Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TREBALL FINAL DE GRAU

TÍTOL DEL TFG: Desenvolupament i avaluació d'escenaris de xarxes de sensors per a docència

TITULACIÓ: Grau en Enginyeria Telemàtica

AUTOR: Joan Ignasi Florit Mir

DIRECTOR: Carles Gómez Montenegro

SUPERVISOR:

DATA: 2 de setembre del 2015

Títol: Desenvolupament i avaluació d'escenaris de xarxes de sensors per a docència

Autor: Joan Ignasi Florit Mir

Director: Carles Gómez Montenegro

Data: 2 de setembre del 2015

Resum

En aquest TFG s'habiliten una sèrie d'escenaris de xarxes de sensors d'acord amb el paradigma de la Internet de les Coses, amb l'objectiu principal de generar material que pugui ser emprat en la docència (en particular, en sessions pràctiques de laboratori) en diverses assignatures de grau i màster de l'EETAC.

El projecte constitueix una guia per tal d'habilitar, configurar i experimentar amb els escenaris plantejats. En particular, s'estudien amb detall les possibilitats que ofereixen les implementacions dels protocols dissenyats o adaptats per a aquest tipus de dispositius. Aquests protocols són l'IEEE 802.15.4, l'IPv6, fent servir els mecanismes 6LoWPAN, l'encaminament mitjançant el protocol RPL, i el protocol del nivell més elevat de la pila de protocols, anomenat CoAP.

Els diferents escenaris es configuren en aquest TFG fent servir un sistema operatiu anomenat Contiki que farem córrer sobre les plataformes hardware anomenades Econotag, així com l'entorn software associat que hi permet interactuar des d'un PC. S'ha pogut executar software assignant diferents rols als Econotag, com el de Border Router, CoAP server, o bé actuar com a sniffer per a la visualització i captura dels missatges enviats via ràdio a través de Wireshark en la pantalla d'un PC. Finalment, s'ha comprovat mitjançant experiments l'impacte en les prestacions de les comunicacions que poden tenir alguns dels principals paràmetres de configuració dels protocols a nivell de capa física, d'enllaç i de xarxa.

Title: Development and evaluation of scenarios of sensor networks for teaching

Author: Joan Ignasi Florit Mir

Director: Carles Gómez Montenegro

Date: September 2nd 2015

Overview

In this TFG we have enabled a number of scenarios of sensor networks according to the paradigm of the Internet of Things, with the aim of generating material that can be used in teaching (in particular, to be used in the practical laboratory sessions) in various courses of the degree and also the master, both of the EETAC.

The project consists in a guide to enable, configure and experiment with the scenarios presented. In particular, the project studies in detail the possibilities offered in the implementations of protocols designed or adapted for such devices. These protocols are IEEE 802.15.4, IPv6, using the mechanisms 6LoWPAN, routing through the protocol RPL, and the higher-level protocol of the stack, called CoAP.

Different scenarios are configured in this TFG using an operating system called Contiki, which will run on the hardware platform called Econotag, with the associated software environment that enables interaction using a PC. It has been possible to execute software and to assign different roles to the Econotag, such as the Border Router, CoAP server or act as a sniffer to capture and display messages sent via radio through Wireshark on the screen of a PC. Finally, it has been proven by experiments the impact on the performance of the communications that can take some of the main settings in the physical layer, the link layer and the network layer.

Dedicatòria:

Voldria dedicar aquests fulls en primer lloc als meus pares, a qui valoro molt tots els esforços que han fet sempre per mi per a poder pagar-me els estudis. Gràcies per haver-me educat i format, fet créixer com a persona i fill vostre amb els valors que m'heu transmès. Aquesta carrera que vaig començar farà ara ja quatre anys no hauria estat possible sense vosaltres. Moltes gràcies pel suport que m'heu donat i pels ànims que sempre he rebut.

A en Carles Gómez, qui ha estat el meu director de projecte i professor a la Universitat. De qui tindré sempre el bon record d'un professor que m'ha ajudat molt durant la carrera, que sempre s'ha mostrat atent per resoldre els meus dubtes, i qui m'ha anat orientant per fer aquest treball possible.

Gràcies també als companys i companyes que he conegut durant la carrera i amb qui he establert amistat, ells han fet més senzilla aquesta tasca.

A tots, moltes gràcies.
Joan Ignasi.

ÍNDEX

TAULES I FIGURES

CAPÍTOL 1. INTRODUCCIÓ

Aquest capítol és la introducció del projecte. Consta de tres parts: motivació, objectius i estructura del document.

1.1. Motivació

Amb l'auge per les Tecnologies de la Informació i Comunicació actual, sorgeixen noves formes de comunicació i nous tipus de xarxes entre dispositius. A més en la societat, es comencen a incorporar molts dispositius tals com telèfons intel·ligents, tauletes, rellotges, televisions; tot enfocat a funcionar amb una connexió a internet.
Aquests dispositius fan córrer una sèrie d'aplicacions i estan dotats d'unes capacitats de còmput i de comunicació; fent que s'incorporin dintre el concepte de '*smart'*, que es refereix a dispositius dotats d'una certa intel·ligència.

El món es troba cada cop més connectat: les màquines, els objectes, el mobiliari urbà, i tot tipus de processos es comencen a monitoritzar i controlar a distància. Els estem traslladant i incorporant processos tals com recollir dades de temperatures, humitats, pressions; per a tractar aquestes dades, i a partir d'aquí poder actuar en funció dels resultats que obtinguem.

Sorgeix així la Internet de les Coses (*Internet of Things*), un concepte que s'atribueix a Auto-ID Center, fundat el 1999 i que s'inspira en el MIT (*Massachusetts Institute of Technology*). Aquest terme, Internet de les coses (IoT) es refereix a una xarxa formada per objectes que són de la vida quotidiana i que es troben interconnectats.

Els analistes asseguren que en els propers anys es veurà una autèntica invasió de tot tipus de dispositius connectats a Internet. Des de roba, accessoris de moda, vehicles, mobiliari urbà. S'estima que uns 33.000 milions de dispositius disposaran de connexió per al 2020, uns 4.3 per persona.

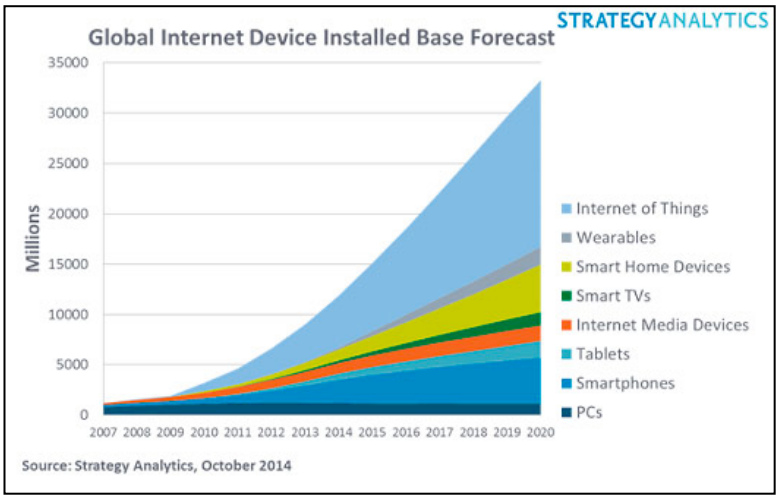


Fig. 1.1 Previsió del nombre de dispositius connectats (dades de Strategy Analytics)

Veiem així el gran creixement de dispositius connectats a Internet de forma global, i el col·lectiu de dispositius que formen la IoT, entrant així en el nou paradigma de la comunicació M2M (Machine to Machine), i on apareixeran nous tipus de xarxes de comunicació entre aquests dispositius.
Tindrem, en aquest context, dispositius sensors que recolliran dades del nostre entorn; els quals serviran per a comunicar les dades, les quals es processaran i s'utilitzaran de tal manera que es millori la qualitat de vida de les persones que componen la societat.

1.2. Objectius

L'objectiu principal d'aquest Treball Fi de Grau és el de desenvolupar escenaris de xarxes de sensors que es puguin emprar en la docència d'assignatures de grau i/o màster de l'EETAC relacionades amb aquest àmbit, com p.ex. les de la temàtica de Smart Cities, Smart Airport, Internet of Things, etc.
Més en concret, en primer lloc es pretén instal·lar una pila de protocols completa basada en IP en la plataforma de node sensor Econotag. Aleshores, s'analitzaran les possibilitats de configuració incloent totes les capes de la pila de protocols, des de la capa física fins la capa d'aplicació.

A continuació, es preveu fer proves amb les configuracions analitzades en diferents escenaris, per tal d'observar els diferents resultats obtinguts i les seves possibilitats per a la docència.

Finalment, es pretén generar el material de suport necessari per a la instal·lació i configuració dels diferents escenaris, que ha de ser d'ajuda per a futurs estudiants i per a professors que imparteixin pràctiques de laboratori en l'àmbit del projecte.

1.3. Estructura del document

Aquest document s'ha estructurat en 6 capítols. En primer lloc hi ha aquest capítol, el primer, que és la Introducció. A continuació, el segon capítol, que és teòric, està dedicat a les xarxes de sensors, aquest inclou una definició, història, característiques, aplicacions, sobre aquests tipus de xarxes. El tercer capítol explica els aspectes teòrics dels protocols relacionats amb la temàtica del TFG: IEEE 802.15.4, 6LoWPAN, RPL i CoAP. El quart capítol està dedicat a explicar quines plataformes de desenvolupament s'han fet servir per aquest projecte, a nivell de hardware i software. El tercer apartat d'aquest capítol tracta sobre les guies respecte de l'entorn software que seran útils de cara als següents capítols. Es veuen també alguns exemples pràctics de programes vistos amb la plataforma hardware, càrregues de programes i altres utilitats que ens serviran per al cinquè capítol que mostra proves i resultats en els escenaris principals del projecte, amb els anàlisis respectius, apartats de configuració, programació, etc.
Finalment, l'últim capítol, el sisè, recull les conclusions i les línies futures que s'extreuen del treball.

CAPÍTOL 2. XARXES DE SENSORS

Aquest es tracta d'un capítol teòric sobre les xarxes de sensors on veurem una primera introducció i definició, la història sobre aquests tipus de xarxes; quins elements les formen; les característiques principals i també quines aplicacions tenen.

2.1. Introducció i Definició

Les Xarxes de Sensors sense fils (*WSN*), estan formades per sensors autònoms distribuïts espacialment, per monitoritzar les condicions físiques o ambientals (p.ex. pressió, so, temperatura, etc.). Aquests sensors col·laboren entre sí per a fer arribar les dades a un destí principal. Actualment les xarxes més modernes són bidireccionals, permeten la comunicació en els dos sentits, permetent a l'usuari de controlar l'activitat dels nodes sensors. Un exemple d'això podria ser la 'smart grid' o xarxa elèctrica intel·ligent; s'envia l'energia elèctrica des del proveïdor als clients tot controlant les necessitats d'aquests, aconseguint un estalvi d'energia, una reducció de costos, més eficiència, etc.

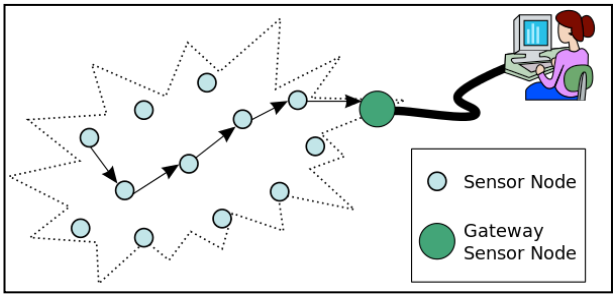


Fig. 2.2 Escenari WSN multisalt

El desenvolupament de xarxes de sensors sense fils originàriament va ser motivat per les aplicacions militars, com la vigilància del camp de batalla. Avui en dia aquest tipus de xarxes s'utilitzen en moltes aplicacions industrials i de consum. Per exemple: monitorització i control de processos industrials, la domòtica, les ciutats intel·ligents, l'agricultura de precisió, l'e-health, etc.

Les xarxes de sensors es construeixen a partir de nodes. El nombre de nodes d'una xarxa de sensors pot anar des d'uns pocs fins a diversos centenars o fins i tot milers de nodes, on cada node està connectat a un o diversos sensors, i també pot incorporar actuadors.

Actualment, en la informàtica i les telecomunicacions, les xarxes de sensors sense fils són una àrea d'investigació que es troba activa amb nombrosos tallers i conferències que s'organitzen cada any (p.ex. IPSN, SENSYS o EWSN), així com a nivell d'estandardització a diferents organismes com l'IETF, l'IEEE, l'ISO o l'ETSI, entre d'altres.

2.2. Història

Com hem mencionat anteriorment, les xarxes de sensors tenen el seu origen en les aplicacions militars.
Un dels predecessors de les xarxes de sensors modernes es considera el *Sound Surveillance System* (SOSUS), que consistia en una xarxa de boies submergides en els Estats Units durant la Guerra Freda, per a poder detectar submarins utilitzant sensors de so.
La investigació en xarxes de sensors, va començar prop de l'any 1980 amb el projecte *Distributed Sensor Networks* (DSN), per part de l'agència militar d'investigació avançada dels Estats Units, *Defense Advanced Research Projects Agency* (DARPA).
Ja en el context actual, segons la Llei de Moore, cada dos anys es duplica el número de transistors en un circuit integrat. La conseqüència directa que té això és que els preus baixen alhora que les prestacions dels dispositius pugen. En 26 anys el nombre de transistors en un xip s'ha incrementat 3.200 vegades.

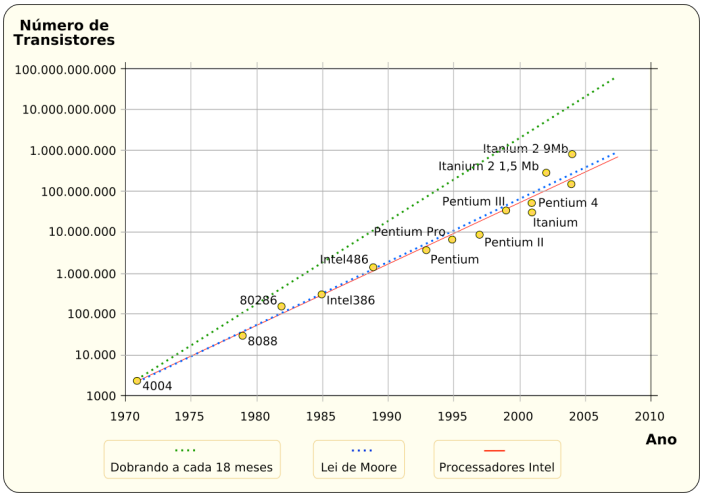


Fig. 2.3 Llei de Moore

Això suposa que els nodes finals rebin l'alies de "*mote*" (de mota com de pols), ja que es preveu que el tamany d'aquests dispositius d'aquí a uns anys si es compleix la llei de Moore, serà realment insignificant.

Un dels projectes pioners en l'àrea de les WSN és va ser el projecte anomenat "Smart Dust", liderat per Kristofer Pister de la Universitat de Califòrnia a finals dels 90, que tenia com a objectiu la construcció d'una xarxa sense fils de minúsculs sensors microelectromecànics MEMS, robots o dispositius que poden detectar diversos tipus de senyals i estímuls. Aquests dispositius van ser anomenats "motes", que havien de tenir una mida d'un gra d'arròs.

Posteriorment, van aparèixer al mercat diverses plataformes de desenvolupament per a nodes sensors, tot i que de dimensions més grans. Un exemple d'aquests tipus de dispositius, motes és el següent, el MICA2DOT (veure la Fig. 2.4).

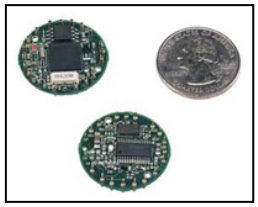


Fig. 2.4 MICA2DOT

2.3. Elements de les WSNs

Els elements que formen les xarxes de nodes sensors (WSN) són els següents:

- Sensors: són els dispositius encarregats de prendre la informació del medi, ja sigui a través de mesures de magnituds físiques o químiques; transformant-la en senyals elèctriques. Típicament, els sensors poden mesurar paràmetres com temperatura, intensitat lumínica, distància, inclinació, acceleració, pressió, humitat, força, etc.

- Nodes sensors (*motes*): Els nodes, coneguts com a motes, són els dispositius principals que formen les xarxes de sensors. Compten amb dispositius sensors, processador, memòria, interfície de xarxa, i també poden comptar amb actuadors. Els nodes sensors prenen dades recollides pels seus sensors, les processen i les envien a través d'alguna interfície de xarxa cap a algun dispositiu que en faci la recollida.

- Estació Base o sink: Recol·lecta dades basant-se en un ordenador comú o bé un sistema encastat. És un component de la WSN amb molts més recursos de computació i energia, solen estar endollats a la xarxa elèctrica, de manera que poden funcionar de manera contínua. Tenen també un major poder de comunicació.

- Tecnologies de xarxa sense fils: es basen en estàndards, com p.ex. el que defineix el protocol IEEE 802.15.4. En el capítol 2 d'aquest treball s'aprofundeix en els conceptes teòrics d'aquest protocol.

2.4. Nodes Sensors

Els nodes sensors de la xarxa estan formats per diverses parts que són les següents:

1) Un transceptor ràdio, que pot portar incorporada una antena interna o bé una connexió a un antena externa.
2) Un microcontrolador.
3) Algun tipus de font d'energia, que normalment és una bateria.
4) Interfícies de sensor.
5) Circuits electrònics per a la interfície amb els sensors.

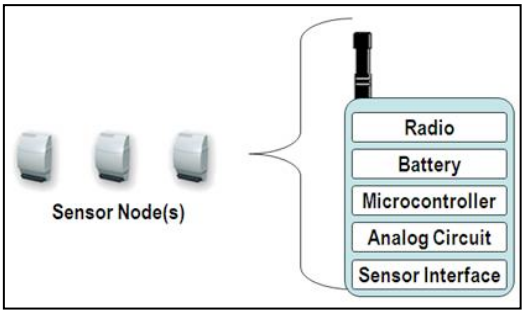


Fig. 2.5 Node Sensor

Pel que fa a les mides d'aquests dispositius, són molt variables, poden anar des de nodes de la mida d'una caixa de sabates, fins a nodes que són de la mida d'un gra de pols (motes).

El cost dels nodes de sensors és igualment variable, els preus poden oscil·lar des d'uns pocs dòlars fins a centenars, depenent de la complexitat dels nodes. Les variacions que hi ha en la mida i el cost dels nodes sensors resulten en les limitacions corresponents als recursos que utilitzin com són l'energia, la memòria, la velocitat de càlcul i l'ample de banda de les comunicacions.

Una de les consideracions tecnològiques principals d'aquests dispositius és la bateria. Per una banda, ens trobem amb que s'ha de considerar el tipus de bateria física que voldrem incorporar al nostre dispositiu. Es consideraran la mida i el pes, així com el cost i la disponibilitat de materials. Una de les opcions més comunes entre els fabricants són les bateries alcalines.

A banda de les anteriors consideracions, també s'ha de tenir en compte, que l'ús del transceptor ràdio consumeix molta d'energia.

Per tal de maximitzar la vida de la bateria, els nodes periòdicament s'encenen per transmetre dades i s'apaguen per a conservar energia. Així la tecnologia ràdio ha de ser el més eficient possible permetent transmetre per una banda, i també entrar en mode *sleep*. Això vol dir que el processador que s'utilitzi també ha de ser capaç de despertar-se i dormir de la manera més eficient possible.

2.5. Característiques de les WSN

Les característiques principals de les Wireless Sensor Networks són les següents:

- Baix Consum: Hi ha una sèrie de restriccions en el consum d'energia pels nodes que utilitzen bateries, o que fan servir algun mètode de recol·lecció d'energia (*energy harvesting*).

- Baix Cost dels nodes: els nodes de les xarxes de sensors, donat que entre les seves característiques destaquen la poca potència que requereixen de transmissió o recepció, els processadors solen ser els d'un tipus de dispositiu senzills, compten amb poca quantitat de memòria RAM i ROM, etc. És a dir, les característiques necessàries per a prendre les mesures

respectives del medi, enviar dades, transmetre, rebre, etc. Tot això fa que els nodes tinguin un cost de fabricació no gaire elevat, cosa que també afavoreix que les xarxes de sensors, les quals solen requerir d'un nombre elevat de nodes sensors; puguin ser desplegades a gran escala, i per tant tinguin una fàcil escalabilitat.

- Mobilitat dels nodes: els nodes sensors són "wireless", per al cas de les xarxes de sensors sense fils; els nodes està alimentats normalment per bateries, que els atorguen mobilitat, i també compten amb una interfície de comunicació ràdio fet que fa que les comunicacions es propaguin sense fils.

- Heterogeneïtat dels nodes: existeixen molts de tipus ben diferenciats de nodes sensors.

- Suport de cicles de Treball: és a dir, dona suport a períodes de baix consum. Quan no tenen dades a transmetre o rebre, els nodes sensors que componen la nostra xarxa entren en períodes en que dormen.
  El percentatge del temps en que els nodes de la xarxa es troben actius es refereix al cicle de treball.

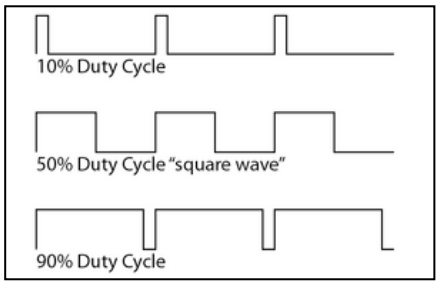


Fig. 2.6 Exemple cicle de treball (Duty Cycle)

- Capacitat dels nodes per suportar dures condicions ambientals.

- Escalabilitat: deguda al baix cost de producció dels nodes, i dels desplegaments en grans àrees.

- Disseny per capes (disseny cross-layer): És un sistema més òptim que no l'enfocament tradicional, permet compartir informació diferent entre les capes. Aquest enfocament permet adaptar-se als canvis ambientals, permet també fer servir la modulació òptima, millora el reniment, la velocitat de transmissió, eficiència energètica, QoS (Qualitat de Servei)...

2.6. Aplicacions de les WSN

Entre les moltes aplicacions actuals de caràcter civil, que tenen les Wireless Sensor Networks (WSNs) podríem destacar-ne les següents:

- Electrònica de consum: Utilització de xarxes de sensors en els equips electrònics que utilitzem quotidianament. Com per exemple el comandament a distància, controls remots, sensors que mesuren la radiació infraroja amb

diversos usos com el control climàtic de la llar, sensors de gas, mesurar la distribució de temperatura dins un forn, etc.

- Automatització de la llar i domòtica: Actualment s'estan reduint molt les mides actuals dels nodes sensors, a més es despleguen cada cop a una velocitat major, essent de cada vegada més assequibles per a l'usuari mitjà. Tot plegat, juntament amb les prestacions, i la interoperabilitat que ens ofereixen amb Internet els nodes basats en IP, fa que sigui una tecnologia que tindrà molt a dir dintre del camp de la domòtica i el control de la llar.

- Àmbit Industrial: Per al control i monitorització en la indústria (maquinària, equipament, vigilància). S'utilitzen per a l'automatització industrial, per detectar si hi ha alguna avaria en la maquinària, per a optimitzar els processos industrials, etc.

- Agricultura intel·ligent: donat que el sector agrícola haurà d'alimentar una creixent quantitat de persones, es preveu que 9.6 milions de persones a l'any 2050, la producció s'haurà d'incrementar en un 70%. Això es farà mitjançant tecnologia de sensors, es recol·lectarà informació sobre rendiment dels cultius, un mapeig del sòl, aplicació de fertilitzants, informació climàtica i també salut animal (temperatura corporal, activitat, pols, ubicació).

- Seguretat i Vigilància: és probablement l'aplicació que trobem més estesa en l'actualitat. Hi ha multitud d'aplicacions per controlar i monitoritzar la llar, per tal de garantir-ne la seguretat. Inclou dispositius tals com alarmes, càmeres, vigilància, sensors de moviment, sensors de ruptura de cristalls, etc.

- Eficiència energètica: Xarxes de sensors que s'utilitzen per tal de controlar que l'ús que es faci de l'electricitat sigui eficaç.

- Medicina i aplicacions mèdiques: la monitorització i control per temes d'aplicacions mèdiques és un camp molt prometedor. Degut a la reducció en la mida dels nodes sensors, ens podem trobar amb que ofereix una millora en la qualitat de vida dels pacients que hagin de tenir controlades les seves constants vitals, ja siguin les pulsacions, la pressió, el nivell de sucre, etc.

- Automoció: Les xarxes de sensors, utilitzades com a complement per a les càmeres de trànsit. Així informen sobre la situació del trànsit en angles morts que per exemple les càmeres no cobreixen; i també es podria informar als conductors de la situació, en cas d'embús o d'accident, fent que aquests poguessin prendre rutes alternatives.

- Sensors Ambientals: Controlar l'ambient en àmplies àrees de bosc o d'oceà, seria impossible sense les xarxes de sensors. Aquestes ens permeten controlar múltiples variables com són la temperatura, humitat, detecció de foc (incendis), activitat sísmica, etc. També ajuden als experts a poder diagnosticar i prevenir un problema o urgència; a més minimitza l'impacte ambiental de l'home.

CAPÍTOL 3. PROTOCOLS EN XARXES DE SENSORS

Aquest nou capítol és teòric i té relació amb l'anterior, on hem vist els aspectes principals de les WSN (definició, característiques, aplicacions). En aquest capítol es veuran els protocols estàndard definits per a aquests tipus de xarxes.

3.1. El protocol IEEE 802.15.4

3.1.1. Introducció

L'IEEE 802.15.4 és un estàndard, el qual defineix el nivell físic i el control d'accés al medi de Xarxes sense fils d'Àrea Personal, concretament centrant-se en aquelles que compten amb taxes de transmissió reduïdes (*Low-Rate Wireless Personal Area Networks*).
El grup de treball IEEE 802.15 (*grup de treball de dintre de IEEE 802 especialitzat en WPAN*) és el que s'encarrega del seu desenvolupament. Es divideixen en cinc subgrups, dels quals el grup número 4 és el que porta a terme la tasca de desenvolupament d'aquest estàndard.
El protocol 802.15.4 també és la base sobre la que més endavant es defineix l'especificació de ZigBee, cosa que suposarà una solució completa per a aquest tipus de xarxes, ja que ZigBee s'encarrega dels nivells superiors de la pila de protocols que l'estàndard 802.15.4 no cobreix. 802.15.4 també és la interfície ràdio per defecte de 6LoWPAN i la pila basada en IP.
El propòsit d'aquest estàndard és el de definir els nivells de xarxa bàsics per a donar servei a les xarxes d'àrea personal sense fils (WPAN), centrat en l'habilitació de comunicació entre dispositius ubics amb un cost reduït i baixa velocitat.

3.1.2. Evolució de l'Estàndard.

L'Evolució de l'estàndard queda dividida de la següent forma:

- **802.15.4: WPAN de baixa velocitat**

  IEEE 802.15.4-2003 (Low Rate WPAN): sistemes amb poques dades a transmetre. Es pretén maximitzar la vida útil, amb alimentació limitada (piles, bateries).

  El grup de treball 6LoWPAN del Internet Engineering Task Force (IETF) treballa en mètodes per operar amb xarxes IPv6 sobre la base de 802.15.4, es descriu al RFC 4919.

  Paral·lelament, alguns dels protocols ZigBee es basen en l'especificació produïda per aquest grup de treball.

- **802.15.4a: Alternativa de nivell PHY**

  El que es pretén és permetre comunicacions, i facilitats de localització amb una precisió molt alta, alta productivitat agregada, tot cobrint unes

necessitats energètiques extremadament reduïdes. Es busca també l'escalabilitat de taxes de dades, la distància de transmissió, el cost i consum.

Al març de 2005 es va seleccionar una especificació de base, consistent en dues PHY opcionals que utilitzen una ràdio de pols UWB i tècniques d'espectre de dispersió Chirp (a la banda de 2,4 GHz). La ràdio de pols UWB es basa en la tecnologia UWB de pols continu (*continuous pulsed UWB*, C-UWB) que és capaç de donar les prestacions requerides.

- **802.15.4b: Revisions i millores**

  Aquest grup es va iniciar amb un projecte de realització de millores i aclariments específiques sobre IEEE 802.15.4-2003. Entre aquests objectius es troben la resolució d'ambigüitats i reducció de complexitat innecessària, l'increment de la flexibilitat en l'ús de claus de seguretat, les consideracions per a l'ús de nous rangs de freqüències disponibles i altres aspectes.

  IEEE 802.15.4b es va aprovar el juny de 2006 i es va publicar al setembre del mateix any com IEEE 802.15.4-2006.

- **802.15.4c: Modificació capa PHY per a la Xina**

  IEEE 802.15.4c va ser aprovada el 2008 i publicada al gener de 2009. Aquesta modificació de les capes físiques afegeix noves especificacions en l'espectre de radiofreqüència, per adaptar-se als canvis de normatives que hi ha a la Xina que han obert les bandes de 314-316 MHz, 430-434 MHz, i 779-787 MHz per a l'ús de WPAN (Wireless Personal Area Network) dins de la Xina.

- **802.15.4d: Modificació capa PHY i MAC per al Japó**

  El grup de treball de IEEE 802.15.4d va ser constituït per definir una modificació en l'estàndard existent 802.15.4 de 2006.

  La modificació contempla canvis tant en la capa física com en la de control d'accés al medi que són necessaris per suportar l'assignació d'una nova freqüència (950 MHz - 956 MHz) a Japó, mentre coexisteixen amb altres sistemes de protocols en la freqüència de banda.

- **802.15.4e: Modificació capa MAC per aplicacions industrials**

  La intenció d'aquesta modificació és millorar i afegir noves funcionalitats a la capa MAC, que bàsicament consisteixen en:

  - millorar el suport als mercats industrials
  - permetre la compatibilitat amb les modificacions que es van proposar al WPAN de la Xina.

Les millores més específiques van ser realitzades per afegir salts de canal i una opció d'intervals de temps variables compatibles amb ISA100.11a.

Aquests canvis van ser aprovats el 2011.

- **802.15.4f: Modificació capa PHY i identificació per radiofreqüència (RFID)**

  El grup de treball de IEEE 802.15.4f va ser constituït per definir noves capes físiques sense fils i millores pel que fa a l'estàndard de la capa MAC 802.15.4 del 2006 necessàries en les noves capes físiques per a la identificació per freqüència o RFID bidireccional.

- **802.15.4g: Modificació PHY per Eines de Xarxa Intel·ligents o SUN**

  El grup de treball IEEE 802.15.4g va ser constituït per crear una nova capa física que modifiqui 802.15.4 per proporcionar un estàndard que faciliti a gran escala aplicacions de control de processos com la utilitat de xarxes intel·ligents capaços de suportar geogràficament diverses xarxes amb una mínima infraestructura. Recentment han sorgit notícies sobre l'estàndard de ràdio 802.15.4g.

A part de les modificacions que s'han anat fent per al protocol, també en trobem d'altres que encara estan en curs, o en fase de desenvolupament, són aquestes:

- **802.15.4j** – Medical Body Area Networks
- **802.15.4k** – Low Energy critical infrastructure monitoring
- **802.15.4m** – TV white space
- **802.15.4n** – China Medical band
- **802.15.4p** – Positive train control

3.1.3. Tipus de Nodes

En l'estàndard 802.15.4 es defineixen dos tipus diferents de nodes o dispositius sensors, que són aquests:

- *Full Function Device (FFD)* o dispositius de funcionalitat completa.
  Aquests poden assumir tots els rols a dintre una xarxa: Device, Coordinator i PAN Coordinator

- *Reduced Function Device (RFD)* o dispositius de funcionalitat reduïda.
  Per a dispositius molt limitats en hardware.
  Només actuen com a Device, per comunicar-se només ho poden fer directament a través de FFDs.

3.1.4. Model de Xarxa (Topologies)

Les xarxes de nodes es poden construir com xarxes punt a punt o en estrella. En qualsevol cas tota xarxa necessita que hi hagi almenys un FFD que faci de coordinador de la PAN (*PAN Coordinator*).

La següent imatge il·lustra com són els dos tipus de topologies, per una banda tenim les xarxes punt a punt que poden formar patrons arbitraris de connexions, per l'altra banda també es poden formar xarxes en estrella on el coordinador serà sempre el node central.

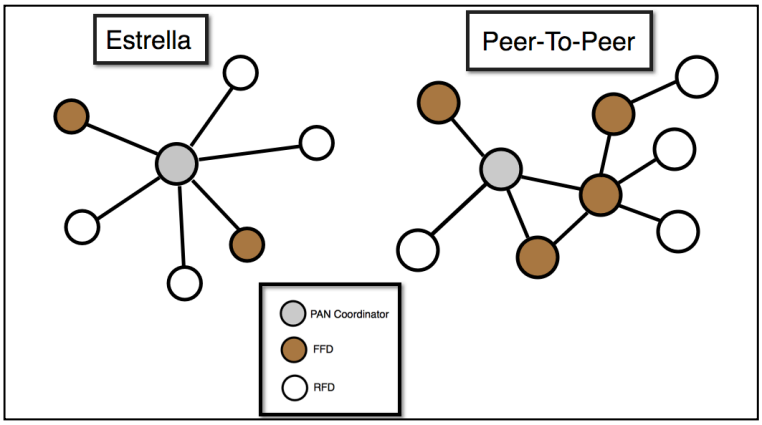


Fig. 3.1 Topologies Xarxes de Sensors

Per crear una xarxa, un nou FFD durà a terme les següents operacions:

- Escaneig de canals
- Escollirà un canal que estigui lliure i un PAN ID que també estigui lliure en aquella àrea.
- El node es converteix en PAN Coordinator (PANC)
- El PANC inicia la transmissió de beacons
- Els altres dispositius FFDs i RFDs s'associaràn a la PAN.

3.1.5. Arquitectura de Protocols

El protocol 802.15.4 defineix una arquitectura de protocols la qual està dividida en nivells, i que està basada en el model OSI. La pila de protocols d'IEEE 802.15.4 és la que podem veure en la Fig. 3.2:

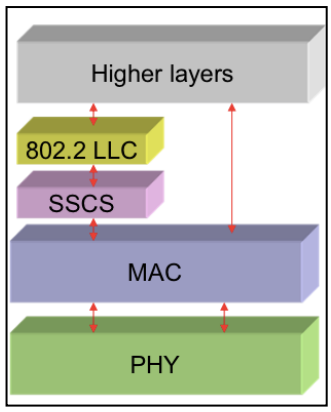


Fig. 3.2 Arquitectura IEEE 802.15.4

L'estàndard defineix els dos nivells inferiors: el nivell físic (PHY) i també el nivell de Control d'Accés al Medi (MAC).
Els dos nivells següents són el Service Specific Convergence Sublayer (SSCS) i el subnivell de control d'enllaç lògic (LLC), que són els que preveuen la interacció amb la resta de nivells superiors de la pila. Aquests dos últims nivells però, no es solen implementar, normalment la capa de xarxa accedeix directament a la capa MAC de 802.15.4.

3.1.6. Capa Física (PHY)

La Capa Física (PHY) ens proporciona la transmissió de dades sobre el medi físic, així com la interfície amb la entitat de gestió del nivell físic. Així aquest nivell controla el transceptor radio, fent la selecció de canals juntament amb el control de consum.
La divisió d'espectre queda així:

- 868-868,8 MHz: A Europa, permet un canal de comunicació en la versió del 2003, en la revisió del 2006: tres canals.
- 902-928 MHz: A Nord Amèrica, fins a deu canals (2003), i fins a trenta en la versió del 2006.
- 2400-2483,5 MHz: A tot el món, fins a setze canals.

Quant a les modulacions que s'utilitzen, la versió original de l'estàndard fa servir DSSS (*Direct Sequence Spread Spectrum*) per eixamplar l'espectre del senyal.

Les modulacions que s'utilitzen són:

- En les bandes de 868 i 915 MHz, s'utilitza la modulació BPSK, i la sensibilitat que proporciona és de -92 dBm.
- En la banda de 2,4 GHz, s'utilitza una O-QPSK, que ens proporciona una sensibilitat de -85 dBm.

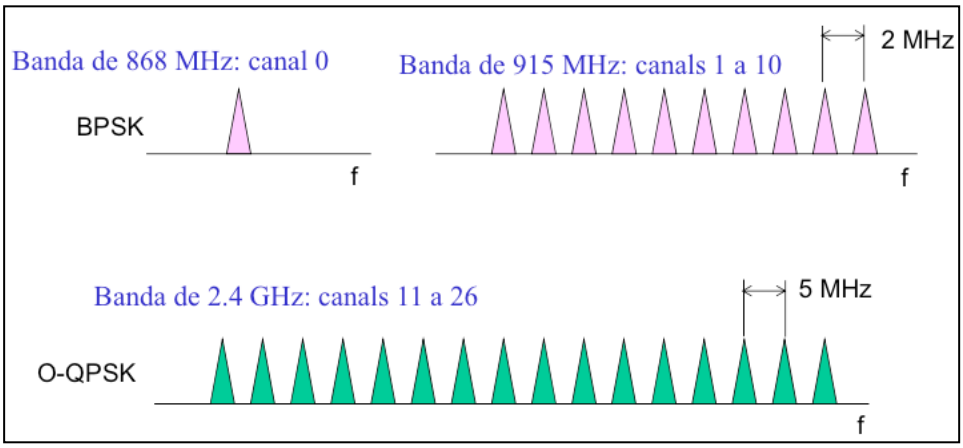


Fig. 3.3 Modulacions, bandes i canals

Quant a velocitats (bit rates) que es defineixen per als estàndards de 2003 i 2006, la cosa queda així:

Taula 2.1. Estàndard IEEE 802.15.4-2003

Banda de Freqüència	Modulació	Bit Rate per Canal (kbps)
868 MHz	BPSK	20
915 MHz	BPSK	40
2,4 GHz	O-QPSK	250

Taula 2.2. Estàndard IEEE 802.15.4-2006

Banda de Freqüència	Modulació	Bit Rate per Canal (kbps)
868 MHz	ASK	250
915 MHz	ASK	259
868 MHz	O-QPSK	100
915 MHz	O-QPSK	250

El format de les unitats de les Unitats de Dades de Protocol de nivell Físic (PPDU), és aquest:

- Preamble de 32 bits: sincronització de trama i de bit
- Strart of Frame Delimiter (SFD): indica la fi del preàmbul
- Frame Length: longitud de la trama
- Physical Service Data Unit (PSDU): "Trama"

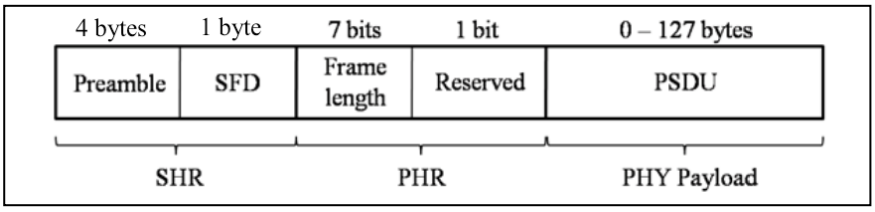


Fig. 3.4 Physical Protocol Data Unit

Existeixen també unes mesures que proporciona el receptor d'una trama de nivell físic, que són:

- RSSI (Radio Signal Strength Indication): per tal d'indicar el nivell de senyal de la trama rebuda.

- LQI (Link Quality Indication): indica la qualitat del senyal corresponent a la trama rebuda. És un valor calculat a partir de la intensitat de senyal rebut, així com el nombre d'errors de la trama rebuda. D'acord amb l'especificació IEEE 802.15.4, els valors mínim i màxim de LQI són 0x00 fins 0xFF, de 0 a 255. L'LQI ens proporciona informació que és útil per a les capes superiors, per exemple per encaminar.

3.1.7. Capa MAC

La capa MAC de 802.15.4 s'encarrega de la gestió de l'accés al medi, la qual pot fer-se mitjançant *beacons*, és a dir amb sincronització, utilitzant l'estructura de supertrama i que fa servir CSMA/CA ranurat amb TDMA. I l'altra opció que ofereix, que és sense beacons, utilitza CSMA/CA.
Una de les altres funcions que desenvolupa la capa MAC de 802.15.4 és la de proporcionar fiabilitat en les comunicacions, fent ús d'ACKs i retransmissions. La fiabilitat és opcional, depenent de l'aplicació serà necessària o no; ja que suposa gastar més recursos d'energia (bateria) i processador per part del dispositiu.
Altres funcions que ens ofereix la capa MAC de 802.15.4 són la d'associació, i també seguretat.
La següent figura ens mostra com és l'estructura de supertrama, la qual està delimitada pels beacons que envien el coordinadors:

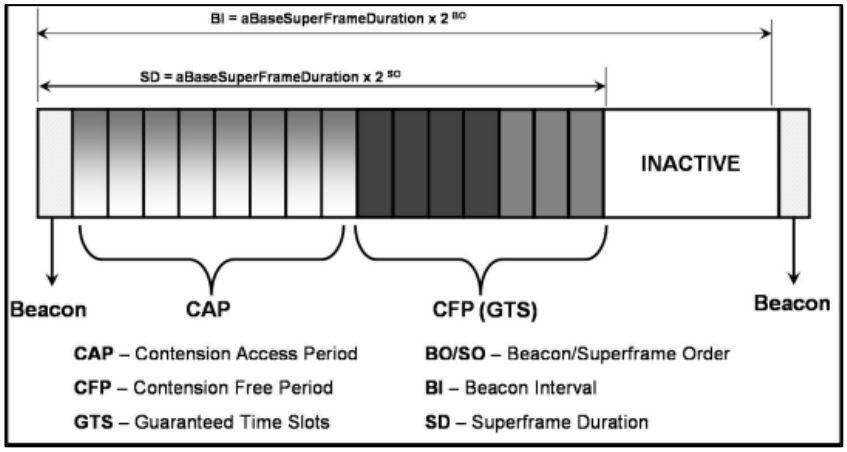


Fig. 3.5 Supertrama

Quant als mecanismes d'accés al medi, en primer lloc tenim que les xarxes amb beacons fan servir CSMA/CA ranurat i TDMA. En canvi les xarxes que són sense beacons fan servir CSMA/CA no ranurat.

El mecanisme TDMA, *Time Division Multiple Acess* és una tecnologia de multiplexació en un medi compartit, permet a un cert nombre d'usuaris compartir el canal ràdio dividint-lo en diferents *slots* temporals. TDMA l'utilitzem en escenaris on hi ha uns requeriments de baix retard, això vol dir que l'accés és garantit ja que es reserva un slot (no es competeix pel medi), resulta més ineficient.

El mecanisme CSMA/CA, *Carrier Sense Multiple Access with Collision Avoidance* és un protocol que utilitzem per evitar col·lisions entre els paquets de dades. Aquest és adient per a un tipus de tràfic sense requeriments estrictes de retard ja que hi ha una competició pel medi compartit. El seu funcionament és el següent: primer tenim un retard inicial aleatori abans de la transmissió, a continuació s'escolta el medi i si està lliure es transmet; en cas contrari es retarda la transmissió un temps aleatori i creixent exponencialment. Ens ofereix

per tant un accés múltiple, i evita la transmissió de trames des de més d'un node simultàniament.

En la Capa MAC de 802.15.4 tenim dos tipus d'adreces: les llargues per una banda (de 64 bits) i que tots els dispositius en tenen una la qual és única; i les adreces curtes (de 16 bits) que les assigna el PAN Coordinator.

3.2. IPv6 i 6LoWPAN

La Connexió de WSNs a Internet ofereix una sèrie d'avantatges com són per una banda la gestió remota de nodes sensors; permetent monitoritzar el seu correcte funcionament; fer actualitzacions de programari (*software*) i també l'accés a bases de dades.
D'una altra banda ofereix també accés remot als nodes sensors, permet accedir a les dades obtingudes pels sensors, i també accedir remotament als actuadors.
A més a més, van anar apareixent un gran nombre de piles de protocols per a WSNs que no oferien suport IP, cadascuna requeria que s'utilitzés un gateway de traducció (PTG).

Els PTGs tenen dos problemes, que són d'una banda la inconsistència entre els dos costats del PTG; i per l'altra banda les limitacions pel desenvolupament d'aplicacions.
Per tant, davant la problemàtica dels gateways de traducció, s'opta per utilitzar IP en els nodes sensors, és a dir utilitzant routers enlloc de PTGs.

El protocol IP ens ofereix una sèrie d'avantatges que són els següents: d'una banda és un protocol que es troba obert i disponible de forma gratuïta; també que és un protocol que és universal tant per a dispositius com per a les aplicacions; a més a més, és un protocol que es troba testejat a gran escala; i finalment diríem que facilita el desenvolupament d'aplicacions sobre aquest protocol.

Del protocol IP, en va sortir més endavant la versió 6 (IPv6), la qual ens permet disposar d'un gran espai d'adreces. Per al cas concret on es vol disposar de multitud de nodes sensors (motes) amb connexió a internet, i direcció IP, això suposa un gran avantatge front a IPv4. La versió 6 d'IP ens permet adreces de 128 bits, cosa que significa poder disposar de 2^{128} direccions diferents, és a dir $3{,}4 \times 10^{38}$. D'altra banda IPv6 també ens permet que les adreces es configurin de forma automàtica (autoconfiguració), a més a més també té un mecanisme de detecció d'adreces duplicades en el moment de fer l'assignació; i també permet el descobriment de routers i prefixos de subxarxa.

En la següent gràfica podem observar quina és la viabilitat d'IPv6 en xarxes de sensors, mirant concretament l'empremta (ocupació d'espai de memòria RAM i ROM) que té aquest protocol:

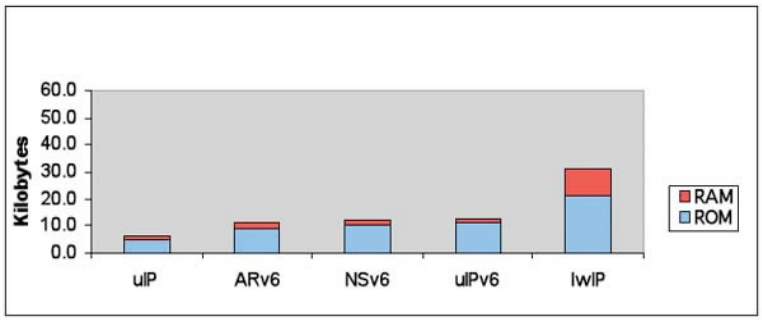


Fig. 3.6 Empremta de memòria d'IP

La posició que ocupa Contiki O.S. en aquesta gràfica és la primera (µIP), i també la quarta (µIpV6), en canvi ARv6 seria per Tiny OS, NSv6 per a Nanostack i per últim lwIP per a lightweigh IP.

Atenent a la Fig. 2.6 veiem com resulta viable instal·lar una implementació d'IP en un dispositiu sensor, els quals solen estar caracteritzats per tenir uns 10 kB de RAM i 100 kB de ROM aproximadament.

Un cop vista l'eficiència, i també que el protocol IP és viable per a les xarxes de sensors, ens apareix un nou paradigma a Internet, com és la comunicació màquina-màquina (abans es feia usuari-màquina, usuari-usuari). Donant peu a la Internet de les Coses, veiem aquí la seva arquitectura de protocols:

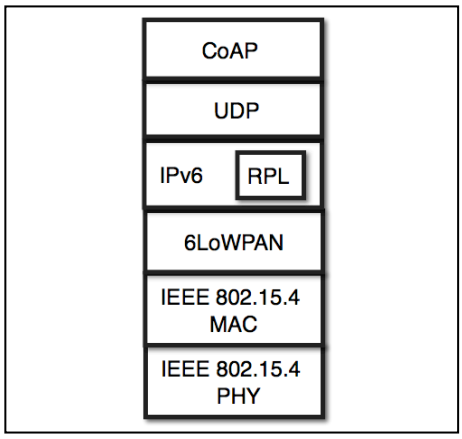


Fig. 3.7 Pila de Protocols Internet of Things

A fi de promocionar la Internet de les Coses, neix la IPSO Alliance que és una organització que promou el Protocol IP per a les comunicacions entre *smart objects* fundada l'any 2008. La revista Time va publicar que la Internet de les Coses era una de les 50 millors invencions de l'any 2008. A més ens trobem amb que la indústria està convergint a fer servir IP: ZigBee, Z-Wave, Wavenis, Bluetooth Low Energy, 802.15.4g.

6LoWPAN (IPv6 over Low Power Wireless Personal Area Networks) és una capa d'adaptació que permet la transmissió de paquets IPv6 sobre xarxes IEEE 802.15.4 (LoWPANs). El seu inici el trobem a l'any 2005, i defineix una sèrie de mecanismes que són els següents: autoconfiguració d'adreces IPv6, un format

específic de trama, la fragmentació de paquets, ofereix també compressió de capçaleres i una optimització del Neighbor Discovery d'IPv6.

3.3. El Protocol RPL

IPv6 Routing Protocol for Low power and Lossy networks (RPL), és un protocol d'encaminament dissenyat per la IETF per a xarxes de sensors, el qual està definit al RFC 6550.
El protocol defineix un graf acíclic dirigit (DAG, *Directed Acyclic Graph*) cap a un node arrel. La posició relativa d'un node en un DAG respecte a la resta ve expressat en termes del valor rank.
L'arrel per defecte té rank=1; el valor del rank creix amb la distància topològica respecte del node arrel.

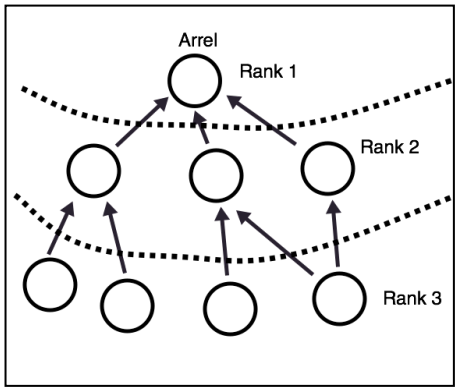


Fig. 3.8 Directed Acyclic Graph (DAG)

El DAG està optimitzat per a l'enviament de dades sensades (les dades que es prenen de l'entorn) fins a un recol·lector o gateway; és a dir en escenaris Multipoint-to-Point (MP2P). També és un esquema adient per a l'enviament de dades des de l'arrel a la resta de nodes; Point-to-Multipoint (P2MP).
En canvi, és subòptim per a l'enviament entre nodes qualssevol, Point-to-Point (P2P), ja que els nodes no es troben directament connectats entre sí, i hauran de pujar la informació fins al node arrel i després que torni a baixar per l'arbre, fent que la comunicació sigui molt lenta. S'ha afegit un mecanisme reactiu per a P2P, basat en AODV; el qual es troba definit al RFC 6997, Reactive Discovery of Point-to-Point Routes in Low-Power and Lossy Networks.
La construcció del DAG ocorre de la següent manera: per començar, els nodes envien missatges DODAG Information Object (DIO) de forma pseudoperiòdica, això està definit al algoritme Trickle (RFC 6206).
Els missatges DIO són enviats per tots els nodes i contenen paràmetres rellevants del DAG: Identificació del DAG, mètrica, temps de vida d'una ruta, etc.
Un node que es vol unir a un DAG ha d'esperar a rebre un DIO o bé enviar un missatge DODAG Information Solicitation (DIS); en base als DIO que rebi, el node aprendrà els seus veïns, d'entre els veïns el node escollirà el seu default parent mitjançant l'Objective Function (OF), el default parent serà el següent salt en el camí fins a l'arrel. L'Objective Function (OF) és una funció que

determina en base a mètriques i restriccions d'una banda el rank d'un node i per l'altra com escollir un default parent.
L'Objective Function Zero (OF0) és un tipus d'OF que indica què es considera bon enllaç (valor 1) i pitjor enllaç (valor 9). Minimum Rank with Hysteresis OF (MRHOF) afegeix histèresi per evitar que les variacions de la qualitat dels enllaços via ràdio es tradueixi en variacions en l'encaminament.

Existeixen dos modes de guardar les rutes: el primer és l'Storing mode, en el qual els nodes emmagatzemen una taula amb rutes, que indica qui és el següent salt per a un destí donat. El segon mode és el non-storing, on els nodes no emmagatzemen taula de rutes; només ho fa l'arrel, que empra source Routing per fer arribar les dades al destí.

Per a formar el DODAG, en primer lloc tenim un node arrel (a) amb rank=1, aquest node envia un missatge DIO al node de sota (b) indicant-li el seu rank. El node b a continuació confirma que ha rebut el missatge DIO i s'anuncia com un possible destinatari enviant un missatge DAO, així ha escollit el seu default parent ja, i també el seu rank=2. A continuació el node b envia un missatge DIO als nodes c i d, informant que el seu rank és 2; ambdós ho confirmaran igual que abans i enviaran un DAO.
En la següent figura podem observar com es forma el DODAG:

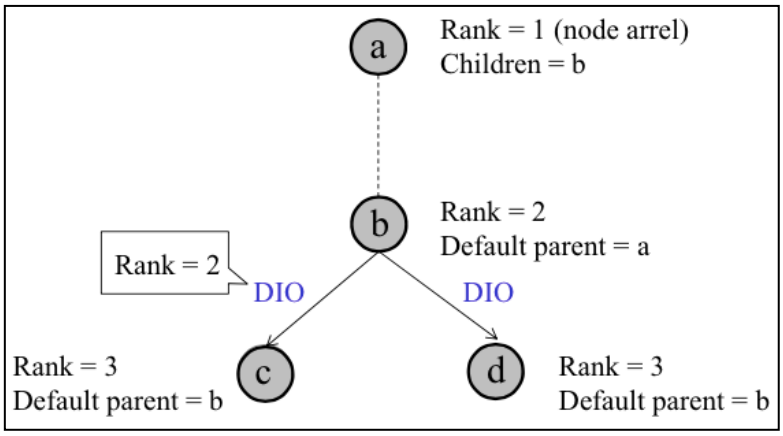


Fig. 3.9 Exemple Formació DODAG

Pel que fa al P2P RPL, el qual s'impulsa des dels àmbits domèstics i també d'edificis, resol d'una banda les rutes subòptimes entre dos nodes qualsevol amb RPL; i també resol alguna possible congestió en la zona que estaria més propera a l'arrel. Es tracta d'un mecanisme reactiu que seria similar a AODV, les peticions de creació de ruta es fan amb DIOs que tenen encapsulat un Route Discovery Option (RDO), la disseminació dels quals es fa mitjançant l'algoritme anomenat Trickle.

Quant a l'algorisme Trickle, el propòsit original d'aquest és la disseminació d'actualitzacions de software per a nodes en una WSN. Està estandarditzat per la IETF (RFC 6206, de l'any 2011). RPL l'empra per regular la transmissió dels DIOs. Aquest protocol té dos mecanismes que són adaptar la taxa d'enviament de missatges segons si un missatge és consistent (amb la informació actual). Així els canvis de topologia es resolen molt ràpidament, ja que la freqüència

d'enviament de missatges és alta; en canvi en una xarxa sense canvis, gairebé no s'envien missatges. L'altre mecanisme de Trickle és la supressió de missatges per limitar la redundància.

Sobre el funcionament de l'algorisme, aquest divideix el temps en intèrvals de durada variable; a cada interval, un node pot enviar un missatge DIO. Trickle determina si l'enviament es pot dur a terme o no.

Els paràmetres que es defineixen a Trickle són els següents:

- I_{min}: durada mínima de l'intèrval (*valor per defecte RPL: 8ms*)
- I_{max}: durada màxima de l'intèrval (*valor per defecte RPL: 2.3 hores*)
- k: constant de redundància (*valor per defecte RPL: 10*)

Variables:

- I: durada de l'intèrval actual
- t: temps de transmissió temptatiu en l'intèrval actual
- c: nombre de missatges escoltats en l'intèrval actual

Per últim, l'algorisme Trickle consta dels següents passos:

1. Assignar a I un valor en el rang [I_{min}, I_{max}]
2. Ressetejar c=0, i assignar a t un valor aleatori en l'interval [$\frac{I}{2}$, I]
3. Incrementar el comptador c quan Trickle sent una transmissió consistent
4. A l'instant t, Trickle només transmet si c < k
5. Dobla la longitud de l'interval cada vegada que expira l'interval actual I
6. Si es rep un missatge inconsistent, es resseteja I=I_{min} i es torna al pas 2

3.4. CoAP

El protocol CoAP, Constrained Application Protocol, està definit al RFC 7252, és un protocol de capa d'aplicació que defineix un model de client/servidor, on els servidors ofereixen representacions de recursos.

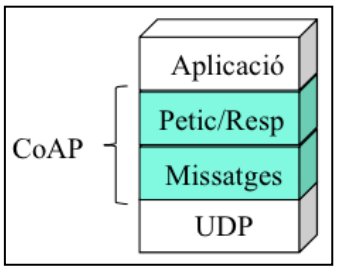


Fig. 3.10 Subcapes de CoAP

Sorgeix degut a que els protocols tradicionals de capa d'aplicació (HTTP, SNMP) no eren adequats per a constrained networks (p.ex. xarxes de sensors)

basades en IP; alguns motius són l'excessiva complexitat de parsejat, els missatges amb un número de bytes excessiu, i que no ofereixen suport als nodes adormits/cicle de treball.

Així el que es va pretendre va ser desenvolupar un protocol d'aplicació que fos adient per a constrained networks, les quals tenen limitacions en ample de banda, taxa de pèrdues, memòria, procés i també energia dels nodes.
El protocol CoAP es basa en l'arquitectura REST, Representational State Transfer, les aplicacions intercanvien representacions de recursos identificades mitjançant URIs, això té l'avantatge que és sense estat. A més ofereix un mapeig senzill amb HTTP, facilitant la interacció entre HTTP i aquest protocol.
CoAP consta de dues subcapes: les peticions amb els mètodes GET, POST, PUT, DELETE.
I els missatges: Confirmable (CON), Non-confirmable (NON), ACK, Reset (RST).

Altres característiques del protocol CoAP són:

- La Fiabilitat en primer lloc que és opcional, algunes aplicacions no en requeriran. CoAP empra UDP a nivell de transport, així que en cas de requerir de fiabilitat s'utilitza un mecanisme de stop-and-wait a nivell de CoAP, s'espera un ACK en resposta a CON abans que expiri un RTO. S'utilitza UDP ja que TCP té una major complexitat, fet que suposa un inconvenient, i també ofereix un rendiment subòptim per a xarxes sense fils. Un altre mecanisme de fiabilitat és el control de congestió, en cas que un missatge CON no és confirmat, es fa una retransmissió, duplicant el valor de l'RTO.

- La Cache, que permet emmagatzemar respostes a peticions proporcionades per servidors, sense haver de tornar a contactar de nou amb el servidor origen.

- El Proxy que fa d'intermediari entre el client i el servidor, fent les corresponents peticions al servidor i responent al client. Permet també emmagatzemar les respostes en una cache, i també que el servidor pugui dormir durant certs períodes de temps.

- Mapeig d'HTTP a CoAP, i viceversa.

- L' Observe Option, que permet als clients que estiguin interessats en la notificació de canvis d'un cert recurs es registrin al servidor mitjançant un GET amb l'opció Observe. I així el servidor transmetrà les pertinents notificacions al client quan succeeixin canvis d'estat.

- Block Option, que permet l'enviament per blocs del payload a transmetre a nivell de CoAP, ja que la mida màxima d'un datagrama UDP són 64 kB, cosa que suposaria un límit en la mida màxima del payload de CoAP. A més, es pot produir fragmentació a les capes inferiors i Block permet evitar-la, fragmentant els payloads de dades d'usuari directament a la capa d'aplicació.

CAPÍTOL 4. PLATAFORMES SOFTWARE I HARDWARE

Amb l'objectiu de desenvolupar els escenaris de Xarxes de Sensors, hem hagut de fer ús d'uns nodes sensors que es connectaran via USB tant per ser alimentats com per a carregar-hi els programes que aquests mateixos executaran. Això ho farem fent servir els PCs que tenim disponibles al laboratori C4-331G de la Universitat, on s'hi instal·larà l'entorn necessari.
Per tal de fer això hem habilitat un entorn virtual per software, com és la imatge d'Instant Contiki que farem anar en els ordinadors del laboratori, i que serà la plataforma que utilitzarem per a programar els dispositius hardware: els nodes Econotag.
A partir d'aquí habilitarem una sèrie d'escenaris, fent també primer proves amb aquest material, per tal de poder veure quines instal·lacions i configuracions ens permeten aquests dispositius. Alhora, generarem material que servirà per a fer pràctiques per als alumnes que estiguin cursant assignatures de l'EETAC de l'àmbit *Smart*, com p.ex. "Xarxes i serveis en Smart Cities I" i d'altres que tinguin una temàtica o continguts similars.

4.1. Entorn Virtual (Software)

Quant a les eines software que hem fet servir per a poder muntar l'entorn virtual adequat per tal de procedir amb el desenvolupament amb els Econotags, ha consistit en fer ús de la imatge Instant Contiki, un Sistema Operatiu Ubuntu basat en Linux que permet instal·lar una pila de protocols basada en IP en nodes sensors (com els Econotag), i que per tant compta amb tots els protocols de la pila basada en IP per a WSNs.
Hem muntat el sistema operatiu sobre l'entorn de Virtualització que ens ofereix VMware.
Per últim també hem fet ús d'algunes llibreries addicionals de Contiki de cara a la implementació dels escenaris. Aquestes són les llibreries libmc1322x del processador mc13224v de Freescale, i també la 6lbr; el seu ús i instal·lació es veuran mes endavant.

4.1.1. Contiki S.O.

Contiki és un sistema operatiu de codi obert que s'executa en petits microcontroladors de baixa potència i permet desenvolupar aplicacions que fan un ús eficient del maquinari, mentre que proporciona comunicació sense fils de baix consum estandarditzat per a una varietat de plataformes hardware.

Contiki s'utilitza en nombrosos sistemes comercials i no comercials, com per exemple: el control de llums del carrer, monitoreig industrial, control de radiacions, sistemes d'alarma i vigilància remota a casa, etc.

Contiki va ser creat per Adam Dunkels l'any 2002 i al seu desenvolupament hi han contribuït equips d'arreu del món com Atmel, Cisco, ENEA, ETH Zurich,

Redwire, RWTH Aachen University , la Universitat d'Oxford, l'Institut Suec de Ciències de la Computació, ST Microelectronics, Zolertia, i molts altres.

Tot i que el sistema operatiu ofereix multitasca i porta incorporada una pila TCP/IP, Contiki només requereix al voltant de 10 kilobytes de RAM i 30 kilobytes de ROM. Això fa que sigui un S.O. molt adient per aquests tipus de dispositius que són de prestacions baixes, i que requereixen de poc consum de recursos i d'energia.

Una instal·lació completa de Contiki inclou les següents característiques:

- Kernel multitasca
- Multitasca apropiativa (pre-emptive multithreading)
- Protothreads
- TCP / IP, incloent IPv6
- Interfície Gràfica d'Usuari (GUI)
- Pantalla remota en xarxa
- Navegador web
- Servidor web personal
- Client Telnet

4.1.2. Virtualització de VMware

VMware és un sistema de virtualització per software. Simula un sistema físic (un ordinador) amb unes característiques i un maquinari determinat. Quan s'executa el programa (simulador), proporciona un *ambient d'execució* similar a tots els efectes d'un ordinador físic (excepte en el *pur accés físic* al maquinari simulat), amb CPU, BIOS, targeta gràfica, memòria RAM, targeta de xarxa, sistema de so, connexió USB, disc dur, etc.

Permet executar diversos Sistemes Operatius dins d'un mateix maquinari de manera simultània, permetent el major aprofitament de recursos.

No obstant això, i en ser una capa intermèdia entre el sistema físic i el sistema operatiu que funciona al maquinari emulat, la velocitat d'execució d'aquest últim és menor, però en la majoria dels casos suficient per usar-se en entorns de producció.

4.2. Hardware (maquinari)

El desafiament principal per a les WSNs és el de produir els nodes sensors que siguin reduïts, en quant a mides físiques i també que el seu cost de fabricació sigui baix. Així, ens trobem amb que hi ha un nombre creixent de petites empreses productores de hardware per a WSN.

Molts dels nodes es troben encara en fase d'investigació i desenvolupament, en particular del seu programari (*software*). Els sensors es fabriquen pensant també amb que utilitzin mètodes de molt baixa potència per a dur a terme les comunicacions ràdio i fer l'adquisició de dades.

En moltes aplicacions, una WSN es comunica amb una xarxa d'àrea local (*LAN*) o xarxa d'àrea estesa (*WAN*) a través d'una porta d'enllaç (*gateway*). El Gateway actua com un pont entre la WSN i l'altra xarxa. Això permet que les dades siguin emmagatzemades i processades pels dispositius amb més recursos, per exemple, en una ubicació remota del servidor.

4.2.1. Plataforma Hardware: Redwire Econotag II

El Redwire Econotag II és un Kit de Desenvolupament MC13224 ARM7 802.15.4 SoC amb JTAG. És la plataforma hardware que es fa servir en aquest projecte. Es tracta d'una placa de desenvolupament per a Freescale MC13224v ARM7, i ràdio 802.15.4. Compta amb un FT2232 que proporciona accés a la interfície JTAG, i UART1. S'alimenta tot a través d'un port USB.
També pot ser alimentat externament amb fins a 16V ja sigui fent servir algun tipus de pila o bateria de cara a aplicacions de baixa potència.
La placa és la que es pot veure a la imatge següent:

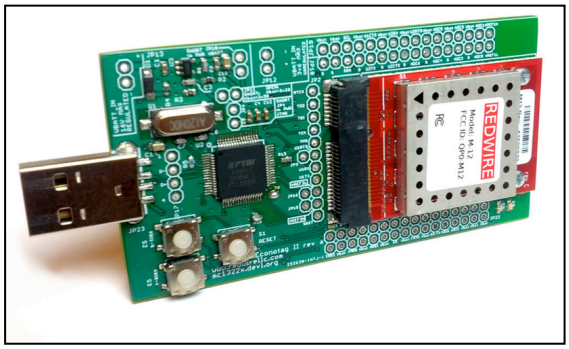


Fig. 4.1 Redwire Econotag II

- Resum de Característiques Principals:

- Microcontrolador Freescale MC13224v ARM7 amb ràdio 802.15.4
- Carregador integrat (UART1, SPI, I2C)
- Antena PCB: a l'aire lliure cobreix aproximadament 500 peus @ 0 dBm. La potència de transmissió màxima és 4.5 dBm
- Cristall de 24MHz
- Botó de restabliment (reset)
- Dos polsadors de propòsit general
- 2 LEDs per a ús general o per RX i TX
- Ponts d'esborrat Flash (també es pot esborrar usant JTAG)
- 36 GPIO treure a 0.1 "pins (inclou tots els pins perifèrics)
- JTAG FT2232H a bord i UART
- Connector USB-A
- *Coixinet* per inductor opcional per convertidor reductor integrat
- *Coixinet* per vidre opcional 32.768 kHz
- Entorn de desenvolupament obert disponible a: http://mc1322x.devl.org
- Disseny de maquinari obert (*open hardware*)

- Microprocessador de l'Econotag: Freescale MC13224v

Fig. 4.2 Freescale MC13224v

És la tercera generació de Semiconductor Freescale 2,4 GHz IEEE 802.15.4. La plataforma fa un salt cap endavant en la capacitat de processament i en la integració de sistemes, alhora que redueix el consum d'energia fins a un 50% respecte les generacions anteriors. El processador és totalment funcional de 32 bits TDMI ARM7 amb una combinació de ROM, RAM i memòria flash integrats.

- Característiques:
  - Baixa potència
    - Consum de corrent típic en mode RX: 21 mA
    - Consum de corrent típic en mode TX: 29 mA
  - 128 KB de flash en sèrie
  - 96 KB de memòria RAM
  - 80K ROM que conté el codi d'arrencada, tots els controladors de dispositius i és compatible amb IEEE 802.15.4 MAC
  - Accelerador MAC (seqüenciador i la interfície DMA)
  - AES de 128 bits per hardware, encriptació / des encriptació amb generador de nombres aleatoris
  - Port JTAG de depuració
  - No requereix de components RF externs

4.3. Guies de l'Entorn, Llibreries i Eines

En aquest apartat es recopilen una sèrie d'eines que s'han fet servir per al desenvolupament d'escenaris, així com les llibreries que hi ha disponibles en repositoris públics d'internet, normalment de github o d'altres portals de desenvolupadors de Contiki. D'aquestes se'n tractarà la seva incorporació a Contiki i també la seva instal·lació; es veurà també com es fan servir amb els dispositius.
Aquest apartat parteix de la base que ja s'ha fet la instal·lació del Sistema Operatiu Instant Contiki. La guia sobre com s'instal·la aquest entorn es troba a l'Annex C.
Les captures que es corresponen amb els programes d'exemple i les llibreries que apareixen en aquest apartat del TFG, les podem trobar recollides en l'Annex E.

4.3.1. Llibreria libmc1322x

La llibreria libmc1322x és la llibreria principal del microcontrolador mc13224v de Freescale, el microcontrolador dels Econotag. La libmc1322x es composa d'un sistema, codi de proves, i també una sèrie d'utilitats per a l'ús d'aquest microcontrolador. A github es troba disponible el seu repositori:
https://github.com/malvira/libmc1322x
En l'apartat de la guia de hardware compatible, s'inclou el dispositiu fabricat per Redwire que fem servir nosaltres en aquest projecte, l'Econotag II.

Quan entrem a la pàgina del repositori, podem comprovar que allà s'explica quins són els passos necessaris que cal seguir per tal d'incorporar la llibreria libmc1322x en el nostre propi codi des del terminal fent servir git. Per a la instal·lació de la llibreria als ordinadors del laboratori 331, hem copiat la llibreria dintre de la ubicació "/contiki-2.7/cpu".

Per tal de poder compilar els programes per a la nostra placa Econotag, ens caldrà disposar d'un fitxer *makefile*, que podrem incorporar als nostres propis programes. Per a incorporar-lo ho farem copiant el fitxer des de la ubicació on es troba dintre de la llibreria:

$ cp libmc1322x/tests/Makefile

Quan volguem compilar un arxiu en .c en el seu binari corresponent per a carregar-lo posteriorment a la placa, haurem de canviar el Makefile per tal que aquest apunti al submòdul *libmc1322x*. Haurem d'afegir a la primera línia d'aquest, on hi posa: '*MC1322X := ..'* ; afegirem *'MC1322X := libmc1322x'*.

És important que un cop ens haguem baixat la llibreria libmc1322x, entrem al path de la carpeta 'tests' i allà fem un *make* per tal de compilar els arxius que hi ha per a les diferents plataformes i generar els binaris corresponents. La placa Econotag ii utilitzarà els arxius binaris que es corresponguin a la placa m12.

4.3.2. Eina Tunslip6 i llibreria 6lbr

L'eina Tunslip té molta utilitat, el que permet és que la mota Econotag connectada al PC pugui enviar les dades rebudes cap al port USB, així es poden veure i també capturar els paquets transmesos i rebuts via ràdio amb el Wireshark que s'executa en el PC.

El protocol SLIP, *Serial Line Internet Protocol*, és un estàndard de transmissió de datagrames IP per a línies sèrie. La seva especificació es troba en el document RFC 1055. Els microcontroladors utilitzen aquest mode d'encapsulació per paquets IP enlloc del protocol punt a punt ja que SLIP fa servir la mida més reduïda.

La llibreria que farem servir per tal d'incorporar l'eina tunslip6 (amb IPv6), la podem trobar en el següent repositori: https://github.com/cetic/6lbr
L'incorporarem a: "/home/user/contiki-2.7/cpu" copiant allà la carpeta 6lbr.
A continuació ens ficarem al path de 'tools' que és on hi ha l'eina tunslip, i la compilarem amb la comanda make de la següent forma:

cd /home/user/contiki-2.7/cpu/6lbr/tools
make tunslip6

Per activar l'eina tunslip a Contiki, obrirem una consola de terminal, on hi tinguem un Econotag connectat. En cas de tenir algun programa o procés carregat a la RAM de l'Econotag i que estigui corrent a la consola; aquesta la pararem fent Control+C.
A continuació ens fiquem al directori on tenim les eines de 6lbr, la implementació de Contiki per al 6LoWPAN Border Router (en el capítol 5 s'entrarà amb més detall en aquest element). És simplement una llibreria addicional que hem incorporat a la Imatge d'Instant Contiki, que hem ubicat a la carpeta "contiki-2.7/cpu/6lbr".
Per executar tunslip, fem:

cd /home/user/contiki-2.7/cpu/6lbr/tools
sudo ./tunslip6 -s /dev/ttyUSB3 aaaa::1/64

On **aaaa::1/64**: serà el prefix de direcció IPv6 que assignem al port slip. Amb això es crea una nova interfície de xarxa "tunX" on X és el número d'interfície d'aquest tipus. Després d'haver fet l'assignació de la direcció IP, podrem fer un ping mitjançant la comanda: **ping6 aaaa::1**

4.3.3. Actualització Programes Contiki

Els canvis en el programari, carpetes i d'altres coses d'importància per al Sistema Operatiu Contiki es publiquen al repositori oficial de Contiki a github. A part de comptar amb la última versió del S.O. actualment la 2.7 és recomanable també actualitzar els fitxers, exemples i configuracions, etc. Per a actualitzar el repositori a la última versió ho farem amb els següents passos:

git clone https://github.com/contiki-os/contiki contiki-git
cd contiki-git
git submodule update --init –recursive

Amb això haurem clonat i actualitzat en el nostre Sistema Operatiu el repositori de contiki, en aquest cas s'anomenarà "contiki-git", i l'ubicarem a dintre de la carpeta "/home/user" juntament amb les altres dues carpetes "contiki" i "contiki-2.7" presents en la carpeta. Serà recomanable seguir actualitzant aquest repositori, ja que s'hi van fent canvis a mesura que els desenvolupadors de contiki publiquen noves actualitzacions dels fitxers.

4.3.4. Connexió dels dispositius Econotags a l'USB Virtual

Per connectar els dispositius Econotag, per mitjà del port USB a les màquines virtuals amb Vmware, i que aquest entorn ens els reconegui, en les màquines del laboratori 331, que executen el Windows; primer connectem els Econotag i després anem al menú:
Player -> Removable Devices -> Future Devices Dual RS232-HS -> Connect (Disconnect from host).

4.3.5. Gestió de Múltiples Econotags en un únic PC

Quant a la gestió de múltiples dispositius Econotag en un mateix ordinador, per al cas d'utilitzar un sistema operatiu basat en Linux, el primer Econotag que connectem serà */dev/ttyUSB0* i */dev/ttyUSB1*.
L'USB0 és per a la depuració port JTAG, mentre que l'USB1 és per programar amb l'mc1322x. Si connectem un segon Econotag a un port USB, aquest s'assignarà de la mateixa manera que l'anterior, l'USB2 que serà per a depuració del port JTAG i l'USB3 que serà per programar la placa.
Per tant veiem com quan connectem el primer Econotag al PC, tindrem que aquest ocuparà els USB0 i USB1; el segon Econotag ocuparà els USB2 i USB3. En resum, quan volem carregar programes a múltiples plaques haurem de tenir en compte el següent: la primera s'assignarà al USB1, la segona USB3, el tercer al USB5, i així successivament fent servir els ports imparells.
Conèixer l'enumeració correcte de ports ens serà útil per tal d'utilitzar les eines mc1322x-load.pl i tunslip6.

4.3.6. Carregar imatges a l'Econotag

En aquest apartat s'expliquen els passos que cal seguir per a carregar imatges a la memòria de la nostra placa Econotag.
Per a fer-ho tenim dues opcions: les podem carregar a la memòria RAM, o bé a la Flash. En el cas de carregar la imatge a la memòria RAM, ens trobem amb que el codi s'anirà executant tant de temps com mantinguem el dispositiu o mota connectada al port USB; un cop desconnectem el dispositiu del port USB, el programa s'esborra. En canvi si carreguem el programa a la memòria flash del dispositiu, això ens servirà perquè el programa s'executi de forma persistent, és a dir el programa es queda "gravat" a la memòria de la placa. Per tant, encara que desconnectem el dispositiu del port USB, el pròxim cop que el tornem a alimentar ja tindrà el programa anterior carregat a la memòria flash.

4.3.7. Carregar un arxiu binari a la RAM de l'Econotag

En primer lloc veurem com es carreguen exemples d'arxius binaris a la memòria RAM de l'Econotag, els fitxers que s'hi carreguen estan prèviament escrits en llenguatge c i els compilarem en arxius binaris per a la seva posterior càrrega a la plataforma hardware, en trobem diversos exemples a dintre dels fitxers del sistema operatiu Contiki.
A continuació descriuré el procés de com carregar un programa senzill de prova, i un cop fet aquest, veurem que es fa de la mateixa manera per a tot els altres programes o exemples.
Les instruccions per carregar l'exemple de hello-world són aquestes:

1. Connectar el dispositiu a la màquina virtual Linux (assegurar-nos que VMware l'ha reconegut, si no haurem de confirmar a l'opció "removable devices" que es trobi connectat)
2. Ens fiquem al path de l'exemple (hello-world):
   cd /home/user/contiki-2.7/examples/hello-world/
3. Per crear el .bin:
   make TARGET=econotag hello-world
4. Per enviar el .bin a l'Econotag:
   cd /home/user/contiki-2.7/cpu/mc1322x/tools

sudo ./mc1322x-load.pl -f /home/user/contiki-2.7/examples/hello-world/hello-world_econotag.bin -t /dev/ttyUSB1

Aquesta comanda:

a) crida el loader: *mc1322x-load.pl*
b) carrega el binari: *hello-world_econotag.bin*
c) i se li indica en quin port USB volem carregar el binari

5. El programa començarà a realitzar operacions, ens sortiran uns punts suspensius en la consola del PC, tot esperant a que premem el botó reset.

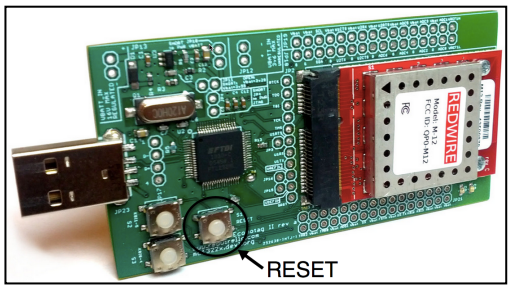


Fig. 4.11 Botó Reset

6. Finalment ens apareix com carrega el programa a la consola, iniciant el procés 'Hello World' en aquest exemple.

4.3.8. Carregar Imatges a la memòria Flash

Per tal de procedir amb la càrrega de fitxers a la memòria flash de l'Econotag, es fa servir el programa "flasher" el qual ve incorporat com a part de la llibreria "libmc1322" a dintre de la carpeta tests.

En principi com hem vist anteriorment, la llibreria la podem trobar a: /home/user/contiki-2.7/cpu/libmc1322x

I si ja hem compilat els binaris amb la comanda make que podem trobar a dintre de la carpeta tests, simplement per a carregar programes a la flash de l'Econotag el que haurem de fer serà cridar el flasher_m12.bin* en el nostre cas per a la placa Econotag II. Això ho farem amb:

cd /home/user/contiki-2.7/cpu/mc1322x/tools

sudo ./mc1322x-load.pl -f /home/user/contiki-2.7/cpu/libmc1322x/tests/flasher_m12.bin -s /home/user/contiki-2.7/examples/hello-world/hello-world_econotag.bin -t /dev/ttyUSB1

A l'Annex F s'hi ha adjuntat una captura amb un exemple de la càrrega d'un programa a la flash.

4.3.9. Esborrar una imatge de la Flash

En alguns casos, com quan per exemple volem carregar un nou fitxer binari a la nostra placa, on ja hi hem carregat algun programa amb anterioritat a la

memòria flash, haurem d'esborrar el programa anterior. També es pot donar el cas que tinguem el dispositiu ocupat amb algun programa a la memòria flash i simplement vulguem esborrar-lo per a tenir la memòria lliure.
Tot això la utilitat bbmc ens ho permetrà, podem trobar aquí el seu repositori: https://github.com/malvira/libmc1322x/wiki/bbmc

L'eina BBMC és part de la llibreria libmc1322x, la podem trobar a:
mc1322x/tools/ftditools

Per a poder fer-la servir necessitem prèviament tenir instal·lat '*libftdi*', versió 0.17 o posterior. En podem comprovar la versió, i després instal·lar:

apt-cache policy libftdi-dev
apt-get install libftdi-dev

A continuació compilarem el programa executant:

cd mc1322x/tools/ftditools
make

Per últim ja tindrem tot instal·lat, i podrem procedir a esborrar la memòria flash d'un Econotag. Primer ens ficarem al path, on executarem:

cd /home/user/contiki-git/cpu/mc1322x/tools/ftditools
sudo ./bbmc -l redbee-econotag erase

4.3.10. Extensió CoAP per a Firefox: Copper (Cu)

Copper és un navegador genèric per a la 'Internet de les Coses' basat en el protocol de la capa d'aplicació CoAP. Segons les estimacions actuals el número de dispositius que s'integraran en les xarxes d'aquest tipus de dispositius serà immens. A més, els sistemes s'optimitzaran per treballar amb energia i amplada de banda limitats; això fa que sigui difícil pels usuaris observar i controlar els dispositius. Mitjançant l'adopció de patrons web, com la navegació, marcadors, i enllaços web ("links"). Copper proporciona una eina de gestió fàcil d'usar per als dispositius integrats en xarxa.
Tot això està integrat en el navegador Firefox en forma d'extensió implementada en JavaScript, que fou testejada en la ETSI 'IoT CoAP Plugtest' de Paris, el Març de 2012. Ens permetrà fer de controlador dels recursos CoAP, fent que ens puguem connectar a una mota que executi l'"Erbium Example Server", que és un codi de servidor de CoAP, des d'un PC.
A mode d'introducció, el seu funcionament és aquest:

- Introduïm una URI CoAP, indicant la direcció IP del nostre servidor de CoAP. Per exemple: coap://[aaaa::205:0c2a:8cdd:e2e6]
- A continuació, clicant sobre el botó 'Discover' podrem veure quins recursos tenim disponibles per a la mota.
- En la localització del recurs, podem utilitzar els botons GET, POST, PUT, i DELETE per a interactuar amb el servidor. Veurem les respostes en el mateix navegador.

- Clicant sobre '  Observe', el que farem serà enviar una petició GET amb la opció Observe. Clicant una segona vegada, esborrarà la subscripció.
- El menú 'Behavior' i les opcions de depuració ('Debug options') ens permeten personalitzar més les sol·licituds i el seu maneig.
- Els recursos que utilitzem més de CoAP es poden marcar com a recursos web normal (Bookmark).

Aquest és el seu enllaç de descàrrega:
https://addons.mozilla.org/en-US/firefox/addon/copper-270430/

4.4. Ús del dispositiu com a Sniffer (Wireshark)

Una altra configuració interessant que ens servirà també per a futurs escenaris, és la utilització del nostre dispositiu Econotag com a Sniffer de paquets, per tal de capturar tràfic i interceptar comunicacions del tipus 802.15.4 o 6LoWPAN.

Aquesta configuració del dispositiu Econotag ens permetrà d'obrir Wireshark capturant des d'un PC, i poder esnifar així paquets de les comunicacions, veure així quins intercanvis de paquets es donen, quin tipus de missatges s'envien els diferents dispositius, i d'altres possible configuracions que ens puguin ser útils de cara a possibles escenaris que vulguem configurar en un futur.

Els passos que he seguit, així com tota la informació sobre com utilitzar el dispositiu com a Sniffer la podem trobar en aquest repositori de github, que forma part de la llibreria libmc1322x d'en Mariano Alvira:
https://github.com/malvira/libmc1322x/wiki/wireshark

El primer pas a seguir serà el d'instal·lar les dependències necessàries de python, que ens serviran per a executar scripts en un futur per tal de fer la captura en temps real. Ho farem executant les següents comanes al terminal:

sudo apt-get install python-serial
sudo apt-get install python-pip
pip install pyserial

A continuació ens assegurem de tenir incorporada la llibreria libmc1322x a dintre de la carpeta cpu de contiki-2.7: "*/home/user/contiki-2.7/cpu/*"

El firmware que utilitzem per a fer de *Sniffer* 802.15.4 amb els Econotags serà el programa 'rftest-rx', el qual és part de la llibreria libmc1322x. El programa es troba ubicat a dintre de la carpeta tests:

cd /home/user/contiki-2.7/cpu/libmc1322x/tests/
make

Un cop compilats els programes de la carpeta 'tests' ens centrarem amb el binari corresponent a la nostra placa 'm12'.
Carregarem per tant el programa 'rftest-rx' a la memòria RAM de la nostra placa Econotag, fent servir el mc1322x-load.pl:

cd /home/user/contiki-2.7/cpu/mc1322x/tools

sudo ./mc1322x-load.pl -f /home/user/contiki-2.7/cpu/libmc1322x/tests/rftest-rx_m12.bin -t /dev/ttyUSB1

Podrem anar incrementant el canal on estem escoltant amb l'sniffer, quan enviem caràcters i el programa no detecti els paquets. Un cop estem en el canal adequat, veurem com els paquets ens apareixen a la pantalla del nostre terminal:

```
channel: 10
channel: 11
channel: 12
rftest-rx --- l rx_time 0x054cd ab ff ff a0 77  07 00 01 00 00 01 00 ff ff ff

1e 40 40 00 00 00 00 00 00 00 0a
aa 00 00 00 00 00 00 00  00 00
```

Fig. 4.12 Canvi de canal Sniffer fins rebre paquets

Un cop comencem a visualitzar els paquets en el terminal on estem executant l'Sniffer, voldrà dir que estem ja en el canal adequat. Ara farem 'Control + C' en el terminal, així el programa se segueix executant.

A continuació, per a fer la captura en temps real amb Wireshark, executarem la següent comanda:

sudo /home/user/contiki-2.7/cpu/libmc1322x/tools/rftestrx2pcap.pl -t /dev/ttyUSB1 | wireshark -k -i -

Se'ns obrirà Wireshark capturant com a usuari 'root', i podrem observar el tràfic 802.15.4 / 6LoWPAN / Zigbee. El resultat final, que veurem de fer servir dispositius com a Sniffers ens servirà per a veure tot tipus de paquets 802.15.4, 6LoWPAN, de CoAP, on intervingui RPL (ICMPv6), etc. Per tant ens serà molt útil a l'hora de fer anàlisis de camps dels paquets, veure formats de trames, i veure que és el que realment està passant en escenaris que es desenvolupin.

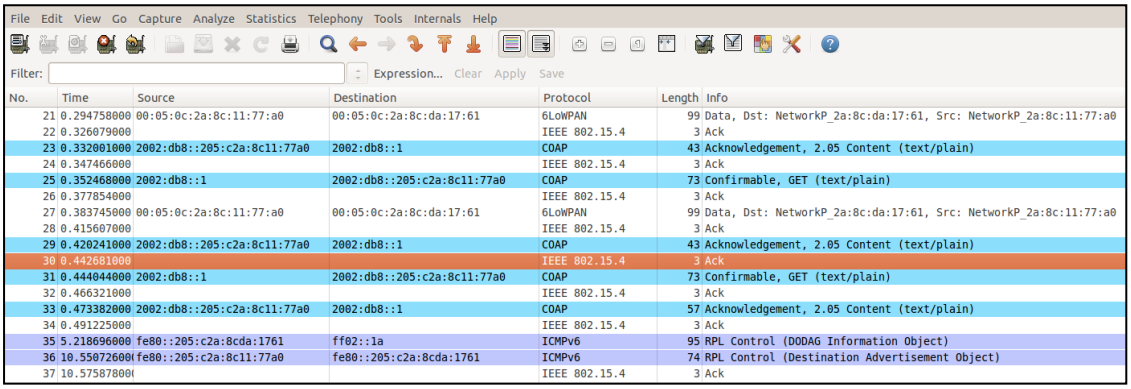


Fig. 4.13 Sniffer Capturant

A l'hora de capturar els paquets rebuts amb Wireshark, podem fer-ho a través d'una captura en temps real utilitzant l'script *rftestrx2pcap.pl*, o també podem fer-ho mitjançant una comana que serveix per capturar els paquets en un fitxer amb extensió *pcap* que podrem obrir amb el Wireshark, és a dir en diferit.

CAPÍTOL 5. ESCENARIS DE PROVES I RESULTATS

Aquest capítol del treball descriu la configuració d'una sèrie d'escenaris que s'han habilitat amb els dispositius Econotag, i que involucren tot el que s'ha vist amb anterioritat sobre la pila de protocols de la Internet de les Coses, des de l'IEEE 802.15.4, arribant fins a les capes més altes, com el routing mitjançant RPL, l'ús d'adreçament amb IPv6 amb 6LoWPAN, i també la capa d'aplicació CoAP. A més a més, es realitzen experiments per verificar les prestacions de diferents aspectes dels protocols en els escenaris construïts.

5.0. 6LoWPAN Border Router: *6LBR*

En els propers escenaris ens apareixerà la figura d'un *6LBR* (o Border Router) com un element que intervindrà en les nostres futures xarxes, i que veurem implementat en un dispositiu Econotag.

Els Border Routers (o routers frontera) ens els trobem als extrems de la xarxa (*edges*). La funció que duen a terme són la de connectar un tipus de xarxa amb una altra. En el nostre cas particular el farem servir per a habilitar un PC per a que pugui interactuar amb una xarxa WSN, tot utilitzant el protocol d'encaminament RPL. Aquest protocol ens ofereix una optimització en els recursos de què disposem: baixa memòria i capacitat de computació; envers el dinamisme que poden tenir aquest tipus de xarxes.
Tota la xarxa de sensors utilitzarà el protocol IPv6, mecanismes com 6LoWPAN s'encarreguen de disminuir la longitud de les capçaleres.

El Border Router disposaria de connexió amb les dues xarxes, i en la WSN seria l'arrel del DAG de RPL que es formaria en la WSN.

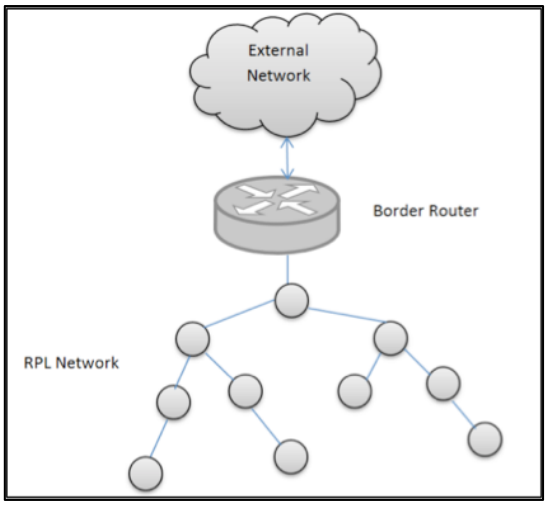


Fig. 5.14 Representació d'un Border Router

En la figura 5.16 de dalt es representa el Border Router com el Router amb símbol de CISCO que es troba en el centre de la figura, és el node arrel de la RPL Network.

L'exemple de Contiki del Border Router el podem trobar en aquest repositori:
https://github.com/contiki-os/contiki/tree/master/examples/ipv6/rpl-border-router

5.1. Primer Escenari: Un sol enllaç ràdio, amb CoAP

5.1.1. Muntatge de l'escenari

El primer escenari comprèn un sol enllaç ràdio 802.15.4 entre dos nodes Econotag. En aquest escenari d'un sol salt tenim els següents elements que intervenen: un PC que estarà connectat mitjançant USB a un dispositiu Econotag, que executarà software per actuar com a Border Router (arrel de RPL), el qual farà de "gateway" entre la connexió USB i l'altra comunicació amb un segon Econotag fent servir el protocol 802.15.4, i que executarà un servidor de CoAP.

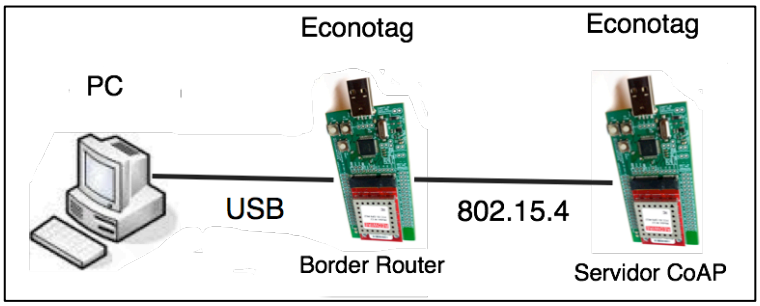


Fig. 5.15 Escenari a un salt

Els passos per tal d'habilitar l'escenari amb una xarxa WSN on intervenen un Border Router (introduït abans) i també un Servidor de CoAP, són els següents:

1. En el primer Econotag carreguem el binari de Border Router.
   Per fer-ho ens haurem de ficar en el path de contiki-2.7 a dintre d'exemples a ipv6 on hi apareix l'exemple del Border Router, i fer el make amb target del nostre dispositiu Econotag, Border Router en aquest cas.

   cd /home/user/contiki-2.7/examples/ipv6/rpl-border-router/
   make TARGET=econotag border-router

   cd /home/user/contiki-2.7/cpu/mc1322x/tools/
   sudo ./mc1322x-load.pl -f /home/user/contiki-2.7/examples/ipv6/rpl-border-router/border-router_econotag.bin -t /dev/ttyUSB3

2. Després farem el tunslip en l'USB del Border Router, parem abans la consola que està executant el Border Router amb Ctrl + C. Ho farem fent servir la llibreria 6lbr que hem incorporat a dintre de la carpeta cpu de contiki:
   cd /home/user/contiki-2.7/cpu/6lbr/tools
   sudo ./tunslip6 -s /dev/ttyUSB3 aaaa::1/64

3. En el segon Econotag carreguem el binari del servidor CoAP:

   cd /home/user/contiki-2.7/examples/er-rest-example/
   make TARGET=econotag er-example-server

   cd /home/user/contiki-2.7/cpu/mc1322x/tools/
   sudo ./mc1322x-load.pl -f /home/user/contiki-2.7/examples/er-rest-example/er-example-server_econotag.bin -t /dev/ttyUSB1

Un cop hem carregat els programes corresponents a les dues motes, podrem observar com es formarà una xarxa entre els dos: un DAG, on el node root serà el nostre Border Router. Podem veure en la Fig. 5.18 els missatges que s'envien corresponents a la formació de la xarxa amb RPL.

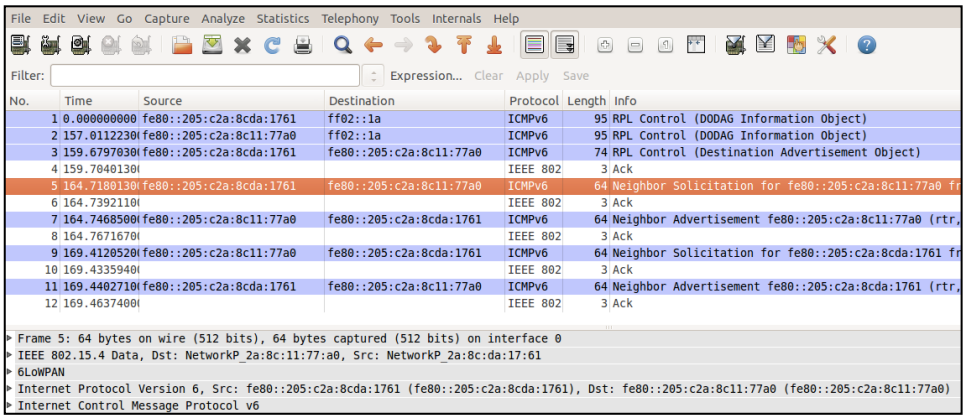


Fig. 5.16 Missatges RPL

A continuació, un cop establerta la xarxa, podrem accedir als recursos web de les dues motes Econotag que intervenen en l'escenari: el Border Router i el Servidor de CoAP.
Tindrem en primer lloc la pàgina web de l'Econotag que fa de Border Router (aquesta s'anomena ContikiRPL), on ens apareixen els nodes veïns i les diferents rutes, com podem veure en la Fig. 5.19. Per a entrar-hi s'ha de posar la url al navegador que es correspon amb la direcció IPv6 del Border Router.
Per trobar la direcció IPv6 del Border Router ens basarem en la seva direcció link-local IPv6:

[**aaaa::205:0c2a:8cfa:9f5b**] (en aquest exemple)

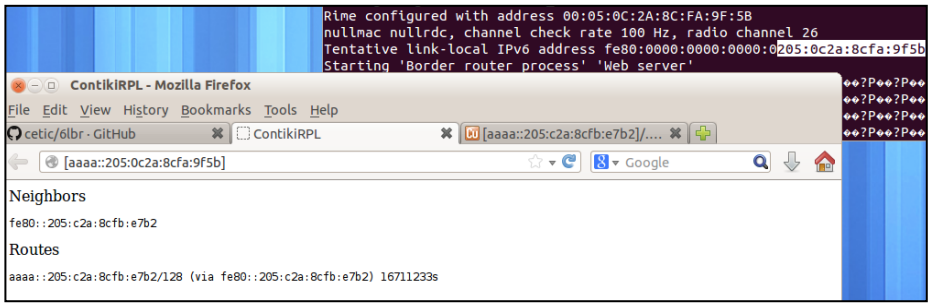


Fig. 5.17 Web del Border Router

A continuació, un cop hem localitzat la direcció IP de la mota veïna, el servidor de CoAP, podrem accedir als recursos que ofereix aquest servidor.
Introduint al navegador: coap://[aaaa::**direcció_IPv6_mota**]

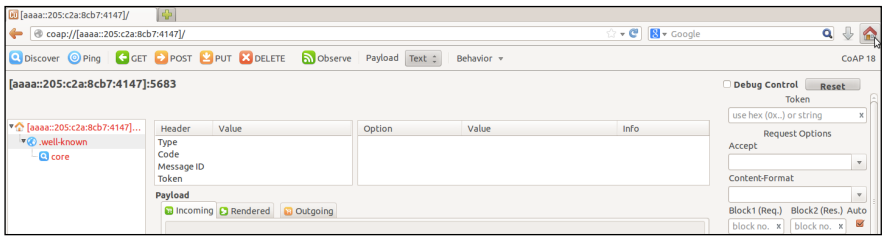


Fig. 5.18 Web servidor CoAP

5.1.2. Diàleg de Formació de la Xarxa RPL

En aquest apartat s'explica com és el diàleg entre els dos nodes que formen el nostre escenari. En aquest cas seria per a la configuració amb la constant **k=2** i amb l'interval **Imin=12**.
Podem veure aquí com la captura comença amb l'enviament del primer DIO per part del Border Router.

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::205:c2a:8c34:b1c7	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
2	3.752006000	fe80::205:c2a:8c8c:4a92	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
3	4.070010000	fe80::205:c2a:8c8c:4a92	fe80::205:c2a:8c34:b1c7	ICMPv6	74	RPL Control (Destination Advertisement Object
4	4.094121000			IEEE 802	3	Ack

Fig. 5.19 Diàleg formació de xarxa RPL

La prova ha consistit en primer lloc en arrencar un dispositiu que faci de Sniffer per capturar els paquets, després s'arrenca el Border Router que serà el node arrel, un cop s'hi activa tunslip. A partir d'aquest instant el Border Router començarà l'enviament de missatges DIO. Gairebé en el mateix instant en què s'ha arrancat el tunslip, s'arrenca també el segon node de la topologia, que serà el servidor de CoAP.

Així es veuen tots els missatges del protocol RPL que s'envien els dos nodes, fins a quedar configurats formant part d'una xarxa. Un cop arrencat el segon node, i amb la recepció del DAO i la confirmació amb l'ack, es visualitza en la pàgina del Border Router (ContikiRPL) el prefix de la xarxa que anuncia el node veí, al qual el Border Router pot accedir-hi a través de la direcció Link Local del Servidor de CoAP.

Els passos que segueixen els dos nodes per configurar-se serien aquests:

- El primer node, el Border Router i l'arrel, envia un missatge DIO
- A continuació, la mota envia també un missatge DIO
- La mota respon amb un missatge DAO indicant la xarxa destinació del node últim que hem connectat: aaaa::205:c2a:8c8c:4a92/128

5.1.3. Afegint nodes addicionals al DODAG

Podem afegir a la nostra xarxa 802.15.4 nous nodes que consistiran en diversos servidors de CoAP que es connectaran al Border Router, podent fer les peticions des d'aquestes motes.
Si afegim fins a quatre nodes, on tenim un Border Router i les respectives motes que s'hi connecten, s'anuncien en la pàgina del Border Router tres nodes que són veïns d'aquest i que anuncien tres xarxes diferents. Podem veure en aquesta captura com queda la taula de rutes:

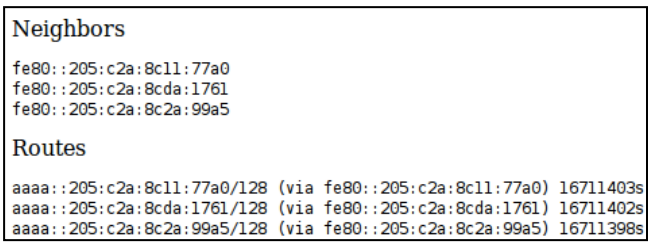


Fig. 5.20 Primer escenari, amb quatre nodes

5.2. Segon Escenari: Escenari Multisalt

El següent pas ha estat, el de configurar un escenari que comporti una xarxa sense fils 802.15.4 de tipus multisalt en el mateix laboratori C4-331G.
Aquesta involucrarà els elements que es poden veure il·lustrats en la següent imatge: tindrem un Border Router, i diverses motes connectades que formaran una xarxa WSN que es configurarà fent servir RPL.

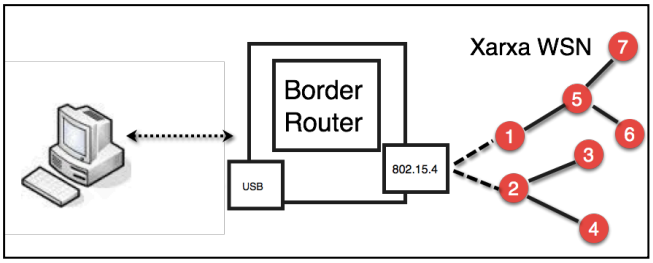


Fig. 5.21 Escenari Multisalt

Per tal de configurar-la, hem fet modificacions en la potència de transmissió, reduint-la i fent que no tots els nodes tinguin una visió directa entre ells, de manera que hi hagi d'haver més salts en les comunicacions que abans eren amb visió directa entre els nodes que formen el nostre escenari, formant així una xarxa de tipus multisalt.
Per tal de formar la xarxa multisalt, les motes es configuren fent servir el protocol de routing RPL que ja hem analitzat anteriorment.
Un diagrama de com s'ha fet la implementació d'aquesta xarxa amb dos salts en el laboratori 331 és el que podem observar en la figura 5.24:

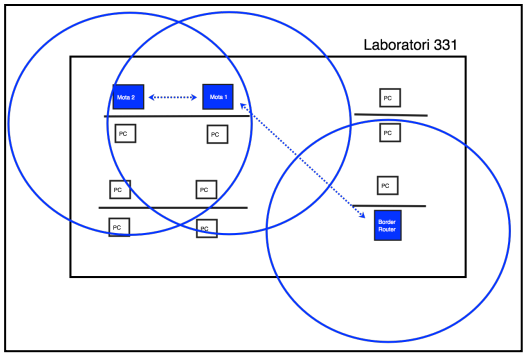


Fig. 5.22 Escenari multisalt laboratori 331

La configuració que s'ha fet servir ha estat la mateixa per a totes les motes:

- Canal: 15
- Potència: 1
- Altres paràmetres: valors per defecte: sense ACKs, mode NullRDC i els paràmetres de RPL per defecte.

En la pàgina web amb informació de RPL del Border Router, la cosa queda de la següent forma. El Border Router veu un node veí directament a través de la link local:

fe80::205:c2a:8c2a:99a5

D'altra banda, el Border Router disposa de les rutes amb destí a les altres motes, a les quals accedeix sempre des del mateix veí.

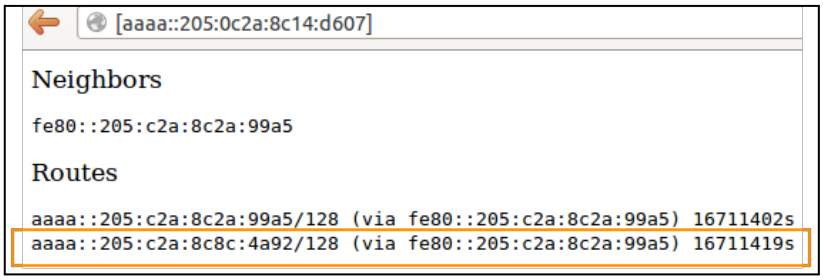


Fig. 5.23 Rutes del Border Router (cas multisalt)

En el laboratori, per les dimensions del mateix, i cobertures ràdio, només hem pogut habilitar un escenari multisalt que involucri dos salts entre els nodes. La idea seria si es disposés de més espai, de poder formar un DODAG amb RPL més extens i que es connectessin les motes per més de dos salts si calgués fins el Border Router, en cas d'haver de cobrir àrees molt grans.

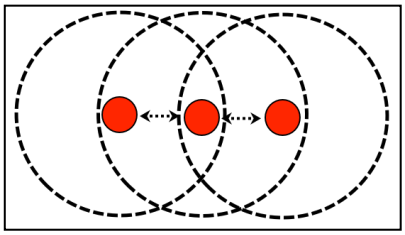


Fig. 5.24 Cobertures dels nodes en una xarxa a dos salts (els extrems no tenen comunicació directe entre ells)

5.3. Mesures de Retard

Quant a les mesures de retard per a arribar als diferents destins des del PC on tenim habilitat l'Econotag que fa de Border Router, hem fet algunes proves fent ping a les diferents motes, tant en l'escenari a un salt com a dos salts, i aquestes han estat les estadístiques finals pel que fa a RTT anotant els valors mitjans (*average*):

Border Router: 20,5 ms

```
--- aaaa::205:0c2a:8c63:dd52 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 19.865/20.503/20.990/0.396 ms
```

Mota 1r salt: 37,4 ms

```
--- aaaa::205:0c2a:8c7f:6c08 ping statistics ---
12 packets transmitted, 11 received, 8% packet loss, time 11025ms
rtt min/avg/max/mdev = 36.507/37.470/38.731/0.829 ms
```

Ping 2n salt: 55,251 ms RTT

```
--- aaaa::205:c2a:8c14:d607 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 54.225/55.251/56.314/0.650 ms
```

Ping (RTT):

Salt 0 : 20,5 ms
Salt 1 : 37,4 ms
Salt 2 : 55,2 ms

Es compleix més o menys el retard que s'afegeix amb cada nou salt que afegim és d'uns 17 ms. En la pàgina de ContikiRPL també apreciem que hi ha un valor temporal major, per a la ruta que és a dos salts.

Neighbors

fe80::205:c2a:8c7f:6c08

Routes

aaaa::205:c2a:8c7f:6c08/128 (via fe80::205:c2a:8c7f:6c08) 16711203s
aaaa::205:c2a:8c14:d607/128 (via fe80::205:c2a:8c7f:6c08) 16711262s

En els apartats següents, s'ha analitzat el que fa referència la configuració de paràmetres del nostre dispositiu Econotag.
Hem dividit els continguts segons els tipus de configuracions fent referència a les diferents capes de la pila de 'Internet of Things' a la qual pertanyen. En primer lloc s'analitza tot el que fa referencia a la configuració de la capa física, a continuació analitzaré la capa MAC, i ja per últim la configuració de RPL.

5.4. Configuració Capa Física

En primer lloc, hem analitzat la capa més baixa de la pila de protocols. Pel que fa a la configuració de la capa física de la placa Econotag, hem fet modificacions en la

configuració inicial que es carrega a la placa pel que fa al processador mc1322x, i que trobem en uns fitxers específics del sistema.

Configuració inicial mc1322x:

```
mc1322x config:
  magic:    1322
  version:  1
  eui:      00050c2a8c8c4a92
  channel:  15
  power:    1
  flags:    fffffffc5
    demod:    1
    autoack:  0
    nvm type: 1
```

5.4.1. Configuració Capa Física: Potència i Canal

Per configurar la capa Física del nostre dispositiu Econotag, cal actuar en els arxius corresponents a la CPU d'aquest. Quan carreguem un programa a la memòria, s'inicialitza la configuració de mc1322x entre la que s'hi inclouen els camps channel i power.

Haurem de modificar els dos fitxers següents que es troben en el directori: "***contiki-2.7/cpu/mc1322x"***

- config.c
- init.c

Farem les següents modificacions ens els camps channel i power simplement, i un cop fet això tornarem a compilar els binaris dels programes que vulguem carregar a la placa. Quan s'inicia la configuració de l'mc1322x veurem com aquesta ha canviat:

config.c

```
void mc1322x_config_set_default(mc1322xConfig *c) {
   nvmType_t type;
   c->magic = MC1322X_CONFIG_MAGIC;
   c->version = MC1322X_CONFIG_VERSION;
   c->eui = 0;
   /* RF_CHANNEL = 26  */
   c->channel = 15;
   c->power = 0x01;
   c->flags.demod = DEMOD_DCD;
   c->flags.autoack = AUTOACK;
   nvm_detect(gNvmInternalInterface_c, &type);
   c->flags.nvmtype = type;
}
```

Prints (config.c):

```
void mc1322x_config_print(void) {
   uint64_t eui64;
   PRINTF("mc1322x config:\n\r");
   PRINTF("  magic:    %04x\n\r", mc1322x_config.magic);
```

```
    PRINTF(" version: %d\n\r", mc1322x_config.version);
    PRINTF("       eui:                %08x%08x\n\r",    (uint32_t)(mc1322x_config.eui>>32),
(uint32_t)(mc1322x_config.eui & 0xffffffff));
    PRINTF(" channel: %d\n\r", 15);
    PRINTF(" power:   %d\n\r", 0x01);
    PRINTF(" flags:   %08x\n\r", mc1322x_config.flags);
    PRINTF("    demod:   %d\n\r", mc1322x_config.flags.demod);
    PRINTF("    autoack: %d\n\r", mc1322x_config.flags.autoack);
    PRINTF("    nvm type: %d\n\r", mc1322x_config.flags.nvmtype);
}
```

init.c

```
    maca_init();
    set_power(0x01);
    set_channel(15);
    set_demodulator_type(mc1322x_config.flags.demod);
    set_prm_mode(mc1322x_config.flags.autoack);
```

5.4.2. Configuració Capa Física: Potència de Transmissió

Es defineix la potència de transmissió a partir de la funció set_power, que introduirem a la nostra funció en hexadecimal, i que podrà oscil·lar entre diversos valors que van des del mínim '0x01' fins el màxim '0x12'.
En aquesta taula se'ns relaciona el nivell de potència que ajustem en la funció set_power amb la potència de transmissió en dBms a la sortida.

Power Level (Hex)	Transmit Power[1] (dBm)	Available for PowerLock[2]
9	-15	No
A	-11	No
B	-10	No
C	-4.5	**Yes**
D	-3	No
E	-1.5	No
F	-1	No
10	1.7	No
11	3	No

Fig. 5.25 Taula de potències de transmissió

Tot i que no està documentat en la figura anterior, els valors de potència arriben fins al 12, que és el màxim i es correspon a una potència de 4.5 dBm. També he provat amb valors inferiors: fins a set_power(0x01) i sembla tenir efecte a nivell de potència rebuda, fent mesures de LQI.

A continuació he fet algunes proves amb el LQI, fent canvis en les potències de transmissió. A partir de dos nodes, un emissor i un receptor separats per una distància fixa (la mateixa sempre durant totes les mesures), d'aproximadament 0,5 metres. He anat canviant els ajustos en la potència dels dos nodes, fixant la

mateixa per ambdós. S'han fet així les mesures de LQI en el receptor indicant els resultats en forma d'intèrval entre els valors màxims i mínims. En la Taula 5.1 podem veure'n els resultats obtinguts:

Taula 5.1. Valors LQI en funció de la potència

Potència (Hex)	Link Quality Indication (Hex)
0x01	0x39 – 0x3f
0x03	0x42 – 0x45
0x05	0x51 – 0x57
0x07	0x5d – 0x60
0x09	0x60 – 0x63
0x0b	0x72 – 0x75
0x0d	0x84 – 0x87
0x0f	0x90 – 0x93
0x10	0x9c – 0x9f
0x11	0x9f – 0xa2
0x12	0xa2 – 0xa8

5.4.3. Configuració Capa Física: Canal Ràdio

La segona configuració de capa física que hem analitzat consisteix en ajustar la configuració del canal ràdio escollit per transmetre.
La configuració del canal ràdio del nostre dispositiu es canvia mitjançant la funció 'set_channel' que cridarem quan inicialitzem un nou programa.
El número de canal que li passem a aquesta funció, mc1322x.config, serà el del canal corresponent de 802.15.4 que van des de l'11 fins el 26. Però en canvi el camp channel enumera aquests des del 0 fins el 15.

- Banda de 2,4 GHz: canals del 11 fins el 26
- Channel: 0 fins el 15

CHANNEL	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EQUIVALENT 2.4 GHz	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

Per a la majoria de proves i exemples de programes que hem fet servir, hem usat el canal 15.
A continuació, per tal de poder veure i comprovar amb un exemple pràctic com els dispositius Econotags no es veuen entre sí quan els configurem en canals ràdio diferents, he carregat un exemple, d'escenari CoAP a un salt fent servir dues motes, on s'ha pogut comprovar que estant una en el canal 12, i l'altra en el canal 15; no s'han arribat a veure entre elles, i el resultat és que el Border Router no anuncia com a veïna l'altra mota, ni tampoc n'anuncia la ruta fins a la mateixa.
En aquest primer exemple, podem veure com les dues motes estan en el mateix canal (channel 12), i es veuen entre sí:

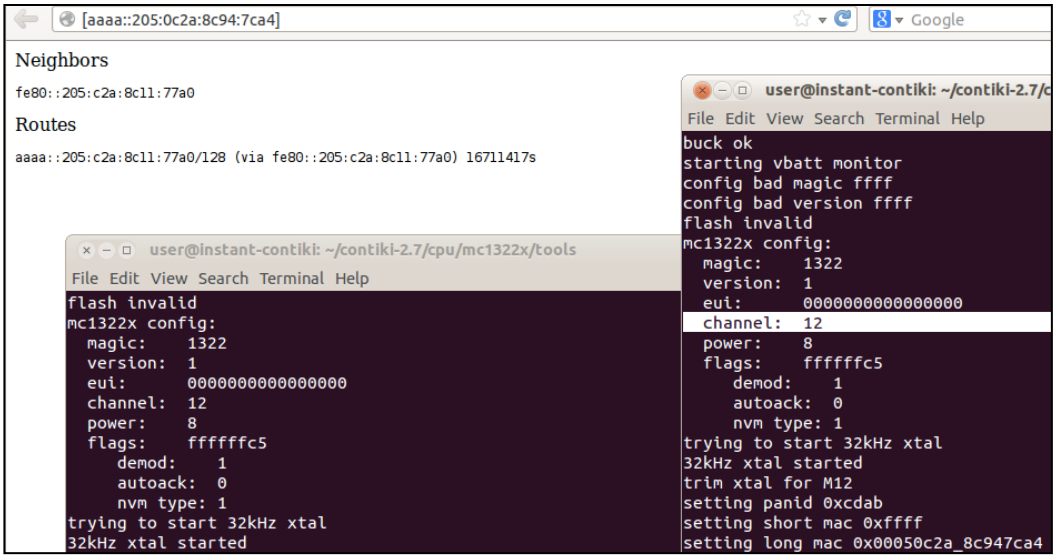


Fig. 5.26 Escenari fent servir el mateix canal

En canvi quan les dues motes es troben en canals diferents, 12 i 15, no es veuen:

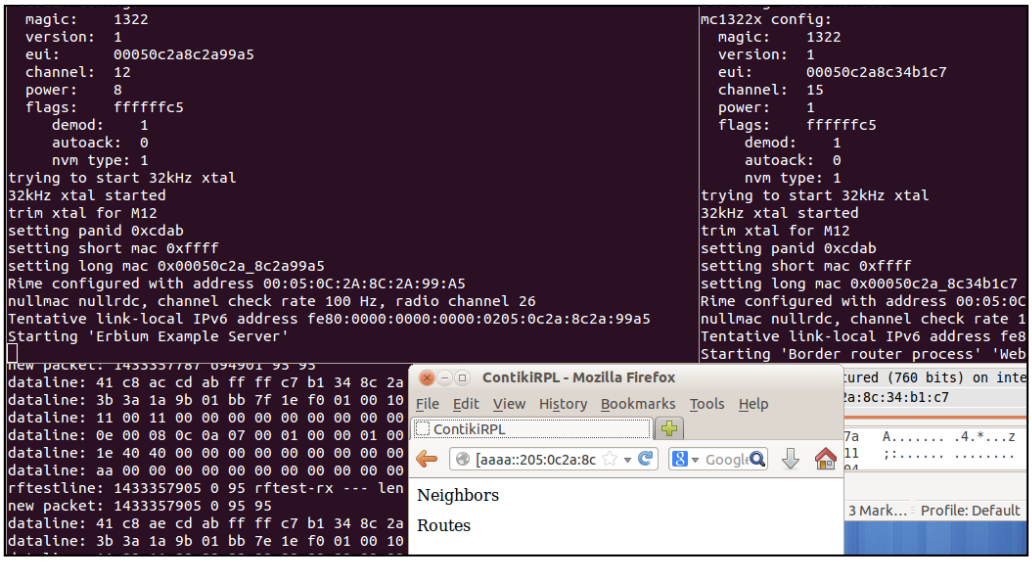


Fig. 5.27 Escenari fent servir canals diferents

El fet de canviar la configuració de canal ràdio de 802.15.4 en els nostres Econotags ens podrà servir per exemple per a poder formar dues xarxes RPL independents entre sí i que aquestes es trobin en canals ràdio diferents; per tant no causaran interferències entre sí.
Això ens podria ser útil de cara a futures proves de docència, o en pràctiques de laboratori, en l'espai del laboratori 331G on ens trobem que per a fer configuracions d'escenaris de RPL, com hi ha poc espai i les potències dels nodes fan que es vegin amb molta facilitat de forma directa, potser es poden treballar escenaris on els grups de treball facin servir diferents canals i així evitar interferències entre ells i poder treballar cada grup amb el seu escenari.

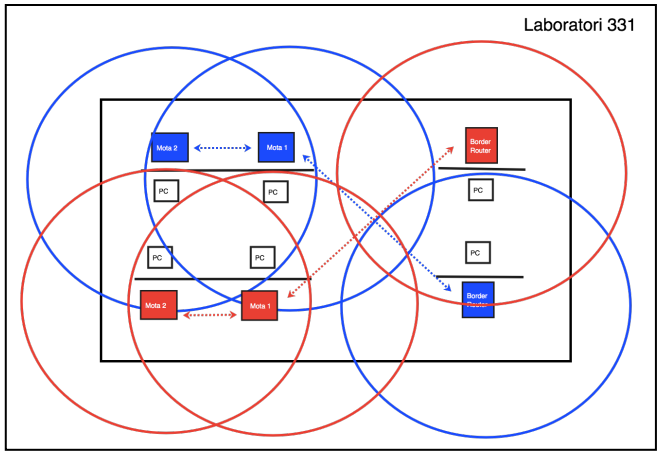


Fig. 5.28 Implementació d'escenaris multisalt en el 331, en canals diferents

5.5. Configuració Capa MAC

Pel que fa a la configuració corresponent a la capa MAC, ens trobem amb que tenim diverses possibilitats: podem definir com serà l'accés al medi (MAC); així com canviar el comportament del transceptor ràdio amb Radio Duty Cycling (RDC) podent definir cicles on el dispositiu es trobarà en mode adormit per a estalviar bateria; per últim podem també definir si volem fer ús d'acks o no.

Els següents fitxers de configuració són rellevants per a la configuració de la capa MAC. El primer es referent a la plataforma, mentre que els altres dos ho són als programes que fem servir per a l'escenari:

/contiki-2.7/platform/econotag/contiki-conf.h
/contiki-2.7/examples/er-rest-example/project-conf.h
/contiki-2.7/examples/ipv6/rpl-border-router/project-conf.h

5.5.1. Modes MAC

A Contiki existeixen dos modes MAC diferents que són el NullMAC i també CSMA. El primer es refereix a que no es controla l'accés al medi, és a dir no s'escolta el canal per a transmetre. El mode CSMA en canvi escolta el canal per tal de prevenir les col·lisions.
He muntat un escenari WSN a un salt amb un Border Router i un Servidor de CoAP per a fer proves amb els dos modes MAC. A partir d'aquí, he calculat l'RTT mitjà que s'obté resultat de fer ping a la mota (Servidor CoAP). Com podem veure en els resultats recollits en la Taula 5.2, s'obté un major retard en fer ping quan es fa ús de CSMA, que no fent servir el mode NullMAC.

Taula 5.2. Paràmetres de la configuració de capa MAC

Valor	Definició	RTT a un salt
CSMA_driver	Utilitza CSMA per accedir al canal	39,1 ms
NullMAC_driver	Transmetre sense escoltar	37,7 ms

En els fitxers de configuració es defineix el mode MAC de la següent forma:

```
#define NETSTACK_CONF_MAC
```

5.5.2. Modes RDC (Radio Duty Cycling)

Els modes RDC definits per Contiki són els següents: NullRDC, SicsLowMAC, X-MAC, CX-MAC, LPP i ContikiMAC.
El mode NullRDC, com indica el seu nom, és el mecanisme que manté el transceptor ràdio sempre encès. El mode SicsLowMAC és un tipus NullRDC que afegeix capçaleres 802.15.4. Pel que fa al mode X-MAC és un mecanisme més antic que ContikiMAC, ofereix menys eficiència, però té uns requeriments temporals menys estrictes, CX-MAC (Compatibility X-MAC) és una implementació de X-MAC amb requeriments temporals inferiors, que funciona en més tipus de ràdios. El mode LPP, Low-Power Probing, és un protocol RDC iniciat en el receptor. I ja per acabar ContikiMAC, que és el mecanisme per defecte, ofereix una molt bona eficiència energètica, però està dissenyat per al transceptor ràdio CC2420.

A continuació s'han elaborat proves amb l'escenari amb un Border Router (fent de node root) i un Servidor CoAP fent servir en cada cas un mode RDC diferent. Les proves amb captures les podem trobar en l'annex G. La Taula 5.3 mostra un resum dels resultats que s'han obtingut:

Taula 5.3. Modes Radio Duty Cycling

Mode RDC	Requeriment de RAM (no figuren unitats*)	RTT salt 0	RTT salt 1	S'han vist les motes?
NullRDC	21	20,4 ms	37,7 ms	Sí
SicslowMAC	25	20,7 ms	34,1 ms	Sí
CX-MAC	115	-	-	Es veu el veí, no s'anuncia la ruta
X-MAC	175	-	-	-
LPP	473	20,4 ms	37,5 ms	Sí
ContikiMAC	984	-	-	No

Com podem veure, els modes LPP i NullRDC donen els mateixos valors de RTT tant per al salt 0 (Border Router), com fins el salt 1 (Servidor CoAP). El mode SicsLowMAC ens ha donat un valor inferior de RTT a un salt respecte el nullRDC.
El mode ContikiMAC sembla ser que no és compatible amb la nostra plataforma hardware, l'Econotag.

Els fitxers de configuració defineixen el mode RDC:

```
#define NETSTACK_CONF_RDC
```

5.5.3. Configuració de l'ús d'ACKs de capa MAC

Una altra de les configuracions disponibles que tenim pel que fa a la capa MAC és l'opció de fer ús d'acknowledgements (ACKs). Per una major fiabilitat, es poden emprar els ACKs. El problema que això suposa és més complexitat pels nodes que ho implementin, major retard i també un menor estalvi de bateria.

En els fitxers de configuració es defineix el mode autoack així:

```
#define MACA_AUTOACK
#define NULLRDC_CONF_802154_AUTOACK
```

Per tal de comprovar els efectes de fer ús d'ACKs o no en els dos escenaris que hem configurat, s'han fet proves amb ping calculant quin RTT mitjà s'obté com a resultat de fer ping tant en l'escenari a un salt com en l'escenari a dos salts. Aquests resultats s'han comparat en els dos casos amb i sense ACKs, i com s'ha pogut veure pels dos escenaris, sempre s'ha obtingut major retard en el cas de fer ús d'ACKs. La Taula 5.4 il·lustra els resultats d'aquestes proves que s'han fet de ping a un i dos salts, i amb i sense ACKs.

Taula 5.4. Configuració amb ACKs o sense ACKs

Valor	Definició	RTT salt 1	RTT salt 2
0	Sense ACKs	34 ms	49,2 ms
1	Amb ACKs	37,9 ms	55,2 ms

5.6. Configuració del protocol RPL

Pel que fa a la configuració corresponent al protocol RPL, el primer pas ha estat localitzar quins fitxers crida contiki, i carrega a la nostra placa quan compilem els programes que es corresponguin als paràmetres i constants d'aquest protocol. Un cop els hem localitzat, s'ha pogut analitzar quines constants i configuracions poden ser interessants.
Els fitxers amb la configuració corresponent al protocol RPL són aquests dos:

contiki-2.7/core/net/rpl/rpl-conf.h
contiki-2.7/core/net/rpl/rpl-private.h

Principalment pel que fa a la part de la configuració dels fitxers associats, hem fet proves canviant els valors de dos paràmetres:

- **L'interval mínim (Imin)**
- **La constant de redundància (k)**

- **Configuració d'Imin**

Imin és la durada mínima de l'intèrval d'enviament de missatges DIO. Es calcula a partir d'una funció exponencial en base 2 on: Imin = 2^n milisegons. Segons l'especificació de RPL, el valor per defecte de l'exponent "n" és de 3,

cosa que significa un intèrval mínim de 2^3 = 8 ms. Aquest valor es considera massa baix quan utilitzem cicles de treball, amb intervals de wake-up que són típicament de centenars de milisegons. Per això, Contiki estableix el valor per defecte n=12; fent que l'intèrval sigui de 2^{12}ms=4,096s.

```
#ifdef RPL_CONF_DIO_INTERVAL_MIN
#define RPL_DIO_INTERVAL_MIN
RPL_CONF_DIO_INTERVAL_MIN
#else
#define RPL_DIO_INTERVAL_MIN 12
#endif
```

1) La primera prova ha consistit en comprovar com és l'enviament de missatges per part d'un node aïllat, el Border Router, que fa de node root un cop activem el tunslip, i comença a enviar missatges DIO. Així hem pogut veure quina és la freqüència d'enviament de missatges DIO amb aquest node, fent servir k=10 en totes les proves (les captures es poden trobar a l'Annex G), i per a tres valors diferents d'Imin. Els resultats obtinguts són els mostrats a la Fig. 5.31:

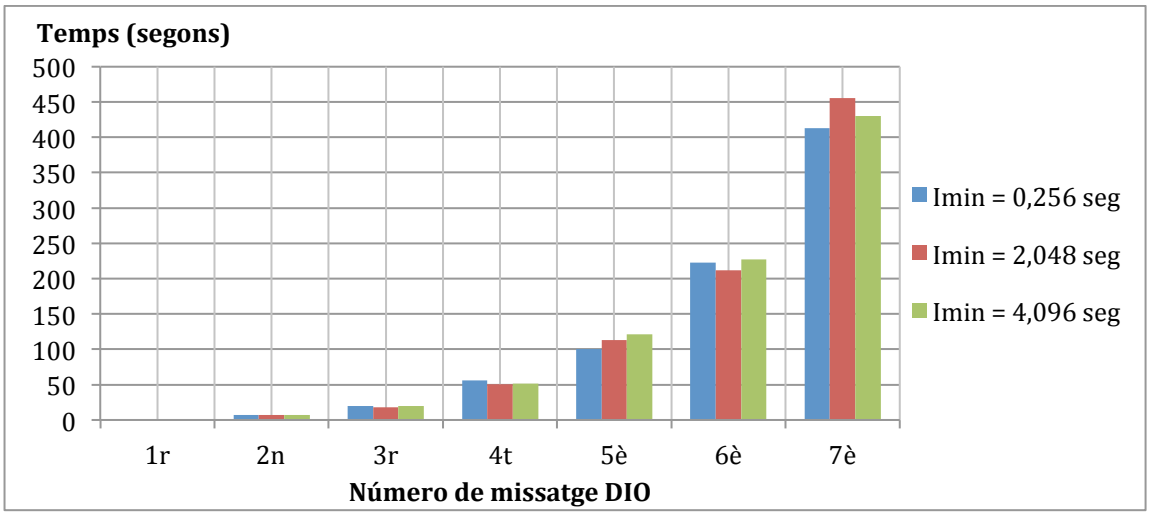


Fig. 5.31 Període d'enviament de DIO's en funció d'Imin, amb un node

Podem observar com els intèrvals entre l'enviament de missatges DIO més o menys sí que es dupliquen, però no apreciem grans diferències entre els enviaments, tot i variar el valor del paràmetre DIO_INTERVAL_MIN (a través del qual es modifica el valor d'Imin). Sí que es pot apreciar un creixement exponencial dels valors temporals amb cada nou missatge DIO que rebem.

2) S'ha fet una anàlisi a continuació fent servir dos nodes: un Border Router i el servidor de CoAP. Configurant primer Imin = 2^{11} = 2 seg; i Imin = 2^{12} = 4 seg. Les captures són a l'Annex G, configuració d'Imin amb dos nodes.

Com hem pogut comprovar, durant el descobriment dels nodes, per a què s'arribin a veure com una xarxa RPL, s'envien una seqüència de missatges de descobriment que segueixen sempre el mateix patró, són aquests missatges en l'ordre que s'indiquen i un darrera l'altre: DIS, DIO, DIO, DAO.

Establint com a referència temporal, l'instant en que es rep el primer missatge DIS a Wireshark, establint aquest com a temps = 0 seg. S'ha calculat quant triguen en cada cas els successius missatges de la seqüència anterior, fent la diferència del temps d'enviament de cadascun respecte el temps del DIS.

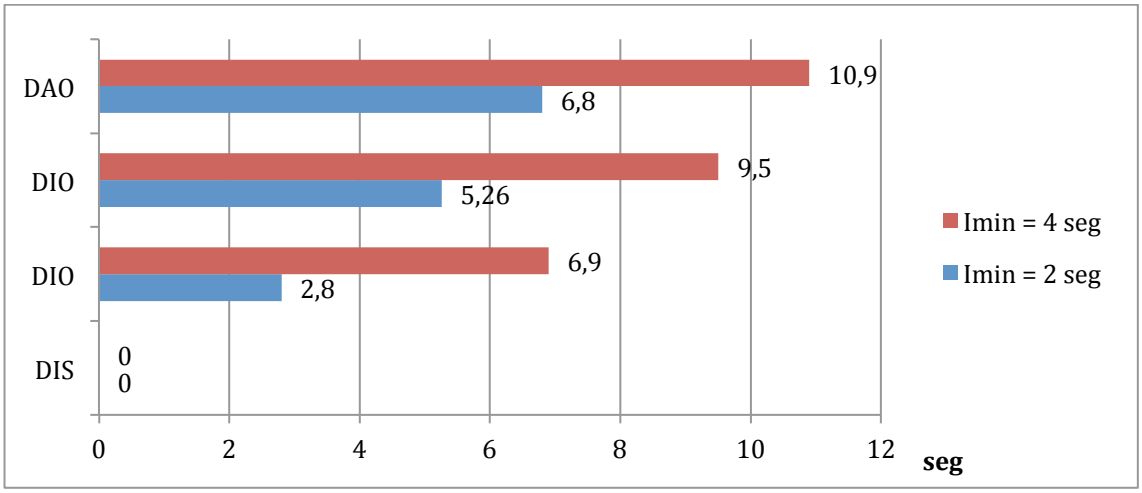


Fig. 5.32 Diferències de temps entre enviament de missatges amb dos nodes, en funció d'Imin

Com es pot veure en la Fig 5.33, els missatges per al cas d'Imin = 2 segons s'envien més freqüentment en tots els casos que els de l'interval de 4 segons. Més o menys es compleix que la freqüència esmentada és el doble, però també s'ha de tenir en compte el factor aleatori de l'algorisme Trickle.

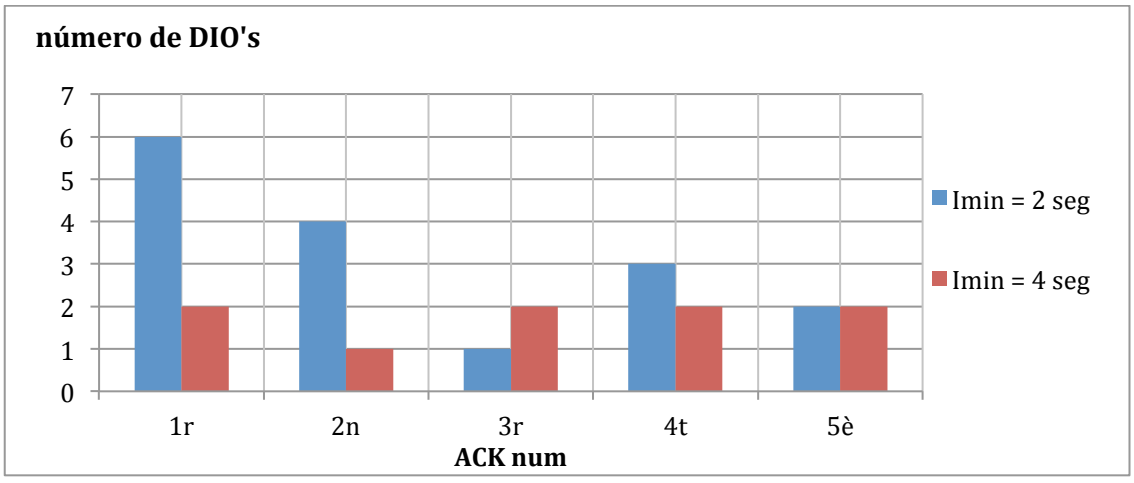


Fig. 5.33 Nº de missatges DIO abans de cada ack, amb dos nodes en funció d'Imin

Com podem apreciar, per al cas de l'interval Imin menor (2 seg), apareixen molts més missatges DIO abans de cada ACK.

3) L'última prova ha consistit en un diagrama comparatiu dels temps de convergència en funció d'Imin. Amb k, amb el seu valor per defecte (k=10).

Els temps de convergència de la Fig. 5.34, s'han calculat mesurant el temps des del primer DIO que envia el Border Router, fins que l'altra mota, el servidor de CoAP, envia el DAO i es confirma aquest enviament amb un ACK.

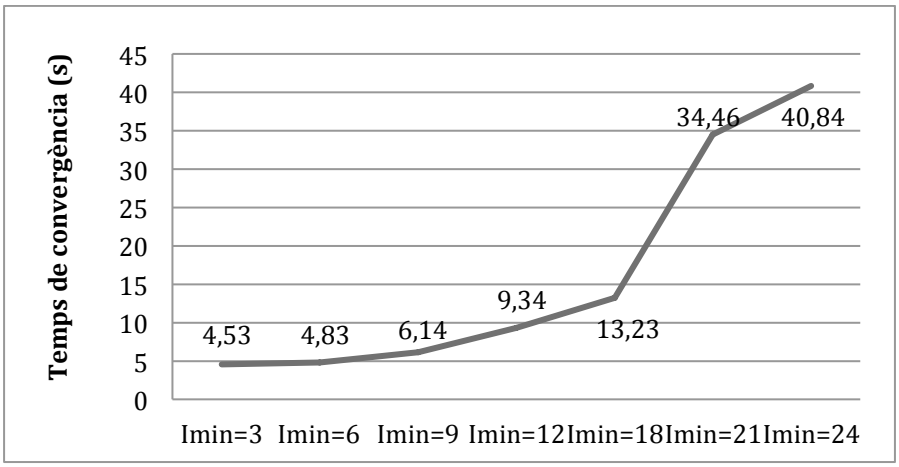


Fig. 5.34 Temps de convergència de la xarxa RPL en funció d'Imin amb dos nodes

Com podem veure en el diagrama, els valors dels temps de convergència de la xarxa RPL, creixen, però no ho fan de forma exponencial com sí que ho hauria de fer l'interval Imin en funció de la configuració, segons la teoria.

- **Configuració constant de redundància (k)**
  La constant de Redundància DIO està descrita en l'RFC 6206. En aquest se'ns suggereix que l'hi assignem un valor per defecte de 10.
  Els operadors de xarxa podran administrar una xarxa de forma més eficient modificant aquest paràmetre, en funció de la implementació específica.

```
#ifdef RPL_CONF_DIO_REDUNDANCY
#define RPL_DIO_REDUNDANCY
RPL_CONF_DIO_REDUNDANCY
#else
#define RPL_DIO_REDUNDANCY 10
#endif
```

La constant k determina el número de transmissions que tindrem en un cert radi de cobertura, durant un cert interval. Tal com està definit en l'algorisme de Trickle, en l'instant t **Trickle transmet si: c < k**.
Per tant un valor elevat de k, farà que hi hagi més transmissions que un valor baix de k.

A continuació, es mostren els resultats de proves que s'han fet amb l'escenari WSN a un salt involucrant dos nodes modificant la constant de redundància k; amb el mateix valor d'Imin = 2^{12} = 4,096 seg, en tots els casos. Així hem pogut veure quina és la freqüència d'enviament de missatges en funció de la k que configurem prèviament.
Els ack's que surten a la gràfica, es refereixen a l'instant temporal que s'ha agafat com a referència, és a dir, com els nodes segueixen una seqüència similar d'enviament de missatges, coincideix que després dels missatges DIO,

envien un acknowledgement com a resposta. Així agafant les quatre primeres respostes, que són els quatre primers missatges ack, s'ha calculat el número de paquets rebuts (missatges), i els valors temporals, a partir d'aquí se n'ha fet el quocient i s'han obtingut els resultats de la gràfica. Els resultats els podem veure en la Fig. 5.35:

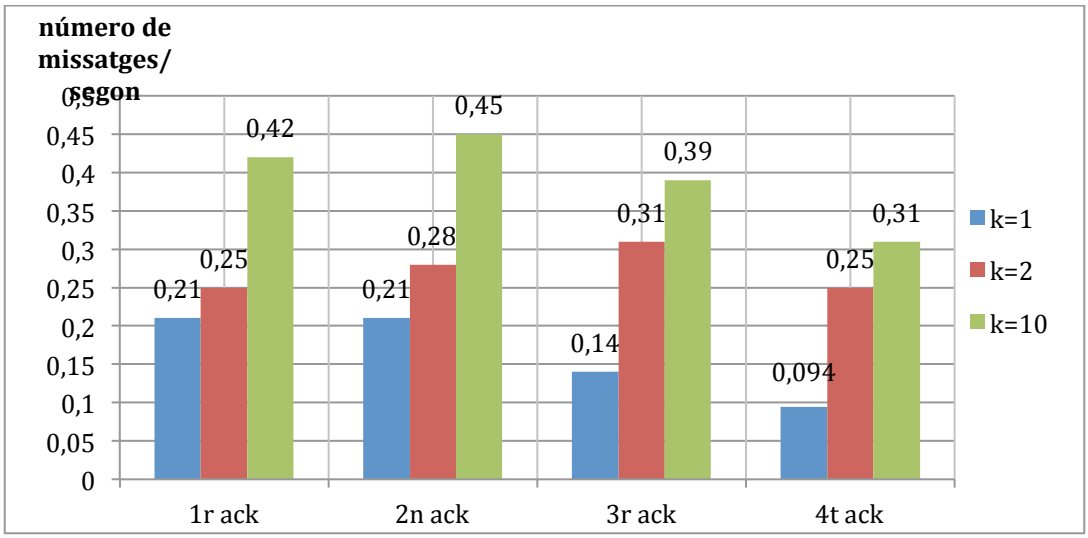


Fig. 5.35 Freqüència d'enviament de missatges en funció de k

Com podem veure, es compleix què la freqüència d'enviament de missatges és major com més gran és el valor de la constant de redundància (k).

La Fig 5.36 mostra els resultats que s'han obtingut de les proves amb l'escenari WSN a un salt amb un Border Router i un Servidor CoAP observant el número de missatges que s'envien, arribant fins a 19 missatges. Això s'ha fet amb diferents valors de k (k=1 , k=2 i k=10), observant amb Wireshark quins temps de recepció tenim per a cada missatge.

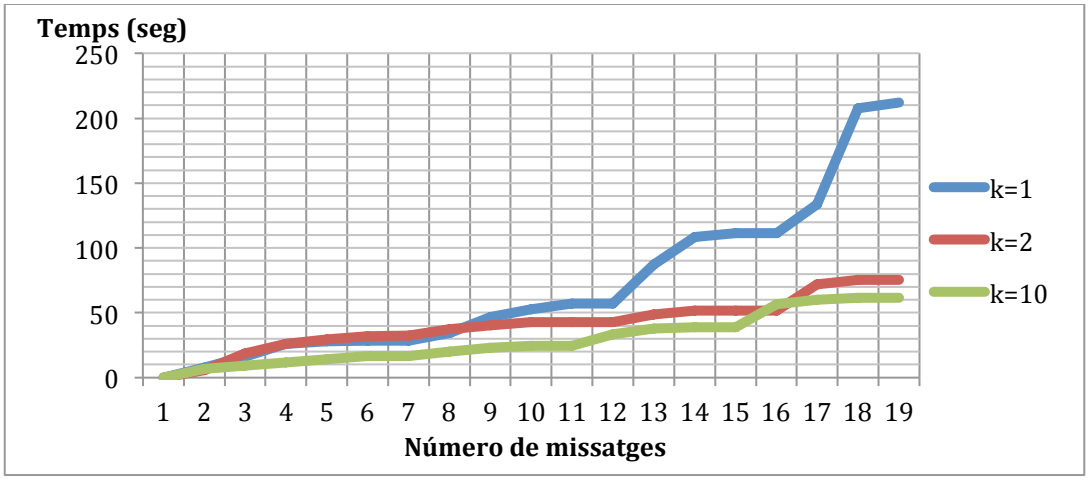


Fig. 5.36 Temps de recepció de missatges en funció de k

Així hem pogut veure com amb valors inferiors de k, l'enviament d'un determinat número de missatges necessita més temps; en canvi, amb valors de k més elevats, aquests temps d'enviament de missatges són inferiors.

CAPÍTOL 6. CONCLUSIONS I LÍNIES FUTURES

L'objectiu principal d'aquest TFG era la generació de material que permetés realitzar pràctiques de laboratori en les assignatures de l'EETAC en les quals s'imparteix docència sobre les xarxes de sensors i la Internet de les Coses. Entenem que l'objectiu ha estat assolit, ja que s'ha pogut realitzar amb èxit la configuració de diversos escenaris de xarxa, utilitzant la plataforma hardware Econotag, que implementa l'IEEE 802.15.4, i que permet executar el sistema operatiu Contiki, que suporta una pila de protocols basada en IPv6, que implementa 6LoWPAN, RPL i CoAP. A més, a més, s'ha pogut analitzar mitjançant experiments l'impacte dels principals paràmetres de configuració dels diferents protocols suportats, i s'ha habilitat la possibilitat d'usar un node a mode de sniffer, permetent visualitzar els missatges enviats via ràdio en la pantalla d'un PC, fet que té un gran valor per a la docència.

Divereses parts d'aquest document constitueixen una guia per a habilitar els escenaris mencionats. La feina feta en aquest treball servirà per a muntar en pràctiques de laboratori amb els escenaris que aquí s'hi han plantejat, amb els diferents rols que es poden assignar als nodes, i fer-ne una anàlisi, així com canviar i variar configuracions, podent compaginar la part de programació i experimental, amb la teoria que hi ha al darrera. Cal mencionar que el software necessari per poder executar els escenaris d'aquest TFG ja ha estat incorporat en les imatges corresponents dels PCs del 331, per tal de poder fer-hi docència ja en el quadrimestre de tardor 2015/16.

Aquest treball donarà als estudiants d'assignatures relacionades amb la temàtica, un impuls per a poder experimentar i veure en la pràctica, l'ús i aplicacions dels protocols que s'estudien en les classes de teoria, veient així com són implementats en la societat actual i futura de la Internet de les Coses i l'IP ubic.

Aquest projecte obre la possibilitat de realitzar moltes hores de sessions de laboratori, muntant, configurant, analitzant els escenaris. De cara a futures investigacions en la línia d'aquest projecte, existeix una eina anomenada Foren6 per a inspeccionar amb un Sniffer el trànsit RPL i poder visualitzar la xarxa que es forma (en l'annex I hi ha més informació). Altres aspectes que es poden seguir investigant són les peticions CoAP de l'Econotag, (es pot provar l'Observe amb el botó, investigar funcionalitats dels sensors), provar els comportaments de Trickle; també veure com actua RPL davant la caiguda d'un node. A més, a github el repositori del Border Router proposa futures investigacions i millores al respecte d'aquest, que potser algun dia es podran implementar també a l'Econotag. En resum, tant Contiki com els Econotags ofereixen moltes possibilitats, i realment es poden seguir fent moltes altres investigacions a banda de les que s'han vist en aquest projecte.

GLOSSARI

6lbr: 6LoWPAN Border Router.

802.15.4: L'estàndard que defineix els nivells físics i de control d'accés al medi de les xarxes inalàmbriques d'àrea personal amb baixes taxes de transmissió de dades (Low Rate WPAN).

ACK: *Acknowledgement*, és un missatge que s'envia per confirmar l'arribada d'un o varis missatges.

AODV: Ad hoc On-Demand Distance Vector Routing. És un protocol d'enrutament per xarxes mòbils ad-hoc (MANET) i altres xarxes ad-hoc.

CSMA/CA: Carrier Sense Multiple Access With Collission Avoidance. Protocol de control de xarxa per evitar col·lisions entre paquets.

DAG: Directed acyclic graph.

DIO: DODAG Information Object.

DIS: DODAG Information Solicitation.

DODAG: Destination Oriented DAG. Topologia que es forma a nivell d'IP, és part del protocol RPL.

FFD: Full Function Device, dispositius que poden dur a terme qualsevol rol a dintre la xarxa.

GUI: Graphic User Interface.

IoT: Internet of Things.

LoWPANs: xarxa de nodes amb interfície IEEE 802.15.4 (RFC 4919, any 2007).

LQI: Link Quality Indication.

MP2P: Multipoint-to-Point.

Neighbor DIscovery: IPv6 Neighbor Discovery (RFC 4861), ofereix descobriment de routers "on-link" i prefixos de xarxa.

P2MP: Point-to-Multipoint.

P2P: Point-to-Point.

PPP: Point-to-Point Protocol.

PAN: Xarxa d'Àrea Personal.

PANC: Coordinador d'una PAN.

PHY: Capa de nivell físic.

RDC: Radio Duty Cycling.

RFD: Reduced Function Device, dispositiu de baixes prestacions que només pot fer de dispositiu final.

RPL: Protocol d'encaminament per a xarxes de sensors. IPv6 Routing Protocol for Low-Power and Lossy Networks.

RTT: Round-Trip Time. Temps que triga un paquet enviat des d'un emissor a tornar al mateix emissor havent passat pel receptor.

SLIP: Serial Line Internet Protocol.

TDMA: Time Division Multiple Access. Tecnologia de multiplexació en un medi compartit.

Trickle: Algorisme que utilitza el protocol RPL per a regular l'enviament dels missatges DIO.

Xarxa ad-hoc: és un tipus de xarxa sense fils descentralitzada, que no depèn d'una infraestructura pre-existent (routers, punts d'accés). Enlloc d'això cada node participa de l'encaminament.

WPAN: *Wireless Personal Area Network*. Xarxes de comunicació entre dispositius molt propers al punt d'accés. Tenen un abans d'uns pocs metres i per a ús personal.

BIBLIOGRAFIA

[1] D. Estrin et al., "Next century challenges: scalable coordination in sensor networks", 1999.

[2] RFC 4919, RFC 4944, RFC 6282, draft-ietf-6lowpan-nd

[3] J. Hui, D. Culler, "IP is Dead, Long Live IP for Wireless Sensor Networks", Setembre 2007.

[4] C. Gomez, J. Paradells, J.E. Caballero, "Sensors everywhere: wireless network technologies and solutions", Fundación Vodafone España, Agost 2010.

[5] IEEE 802.15.4-{2003, 2006}. Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR- WPANs).

[6] IEEE 802.15 Working Group: http://www.ieee802.org/15/

[7] B. Latré et al., "Throughput and Delay Analysis of Unslotted IEEE 802.15.4", Journal of Networks, Vol. 1, pp. 20-28, May 2006.

[8] Contiki: The Open Source OS for the Internet of Things: http://www.contiki-os.org/

[9] Repositori de Contiki: https://github.com/contiki-os/contiki

[10] ContikiProjects: http://sourceforge.net/p/contikiprojects/code/HEAD/tree/

[11] Out in the Open: The Little-Known Open Source OS That Rules the Internet of Things: http://ercim-news.ercim.eu/en76/rd/contiki-bringing-ip-to-sensor-networks

[12] Repositori d'mc1322x: https://github.com/malvira/libmc1322x/wiki

[13] MC1322x Datasheet: http://www.freescale.com/files/rf_if/doc/data_sheet/MC1322x.pdf

[14] A. Dunkels, "The ContikiMAC Radio Duty Cycling Protocol", December 2011

[15] Llibreria 6LBR: https://github.com/cetic/6lbr/wiki

[16] Foren6: http://cetic.github.io/foren6/

ANNEXOS

TÍTOL DEL TFG: Desenvolupament i avaluació d'escenaris de xarxes de sensors per a docència

TITULACIÓ: Grau en Enginyeria Telemàtica

AUTOR: Joan Ignasi Florit Mir

DIRECTOR: Carles Gómez Montenegro

SUPERVISOR:

DATA: 2 de setembre del 2015

ANNEX A: TEORIA SOBRE PROTOCOL 802.15.4

1.1. Capa MAC: Tipus de trames

El format genèric de la trama MAC de 802.15.4 és aquest:

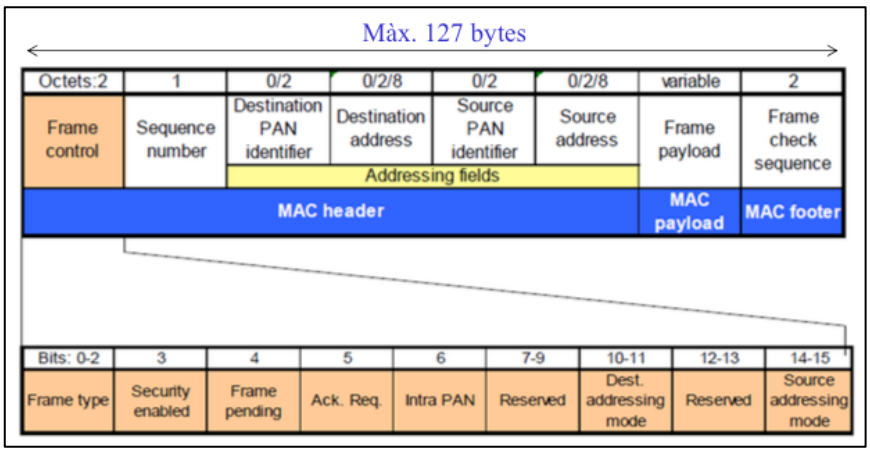


Fig. 29 Trama MAC 802.15.4

Trama Beacon

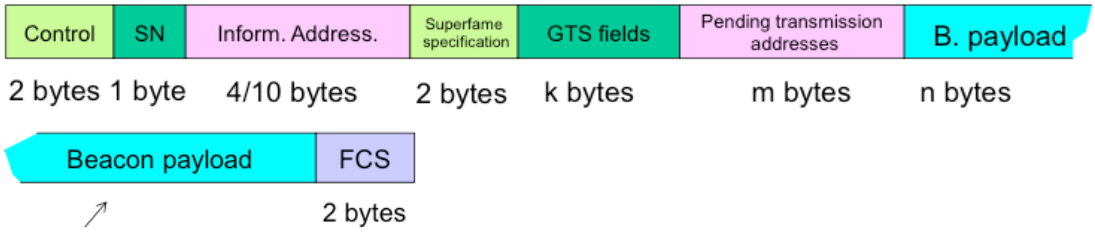


Fig. 30 Format Trama Beacon

Trama de dades

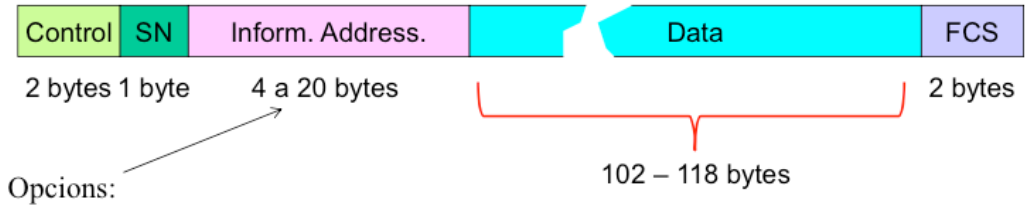


Fig. 31 Format Trama de Dades

Trama ACK

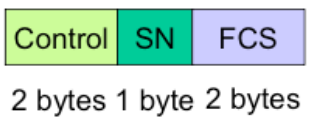


Fig. 32 Format de la Trama ACK

Trama ack vista amb Wireshark:

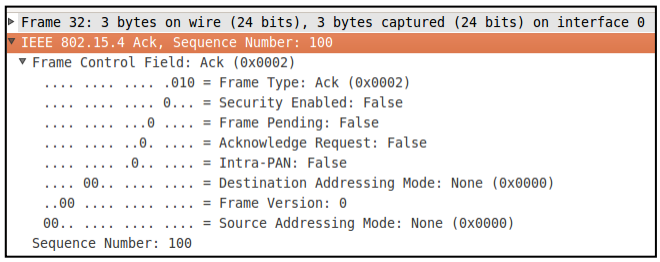


Fig. 33 Trama ACK

1.2. Tipus de Missatges de RPL

Format genèric dels missatges de RPL, s'encapsulen en ICMPv6

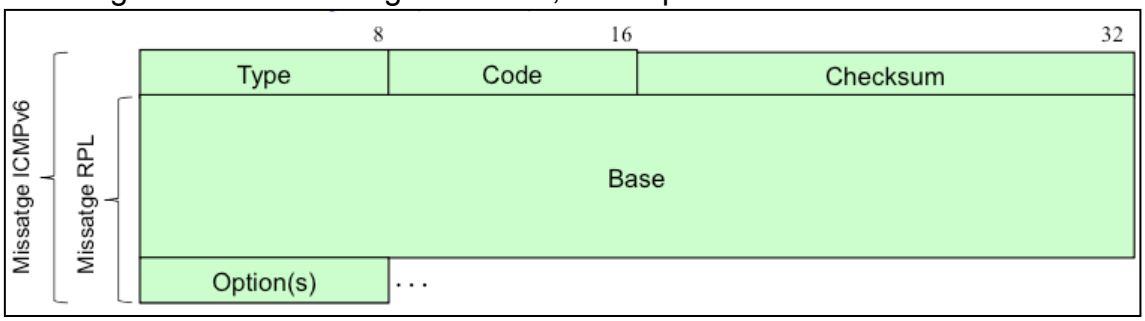


Fig. 34 Missatges RPL

Missatge DIO (DODAG Information Object)

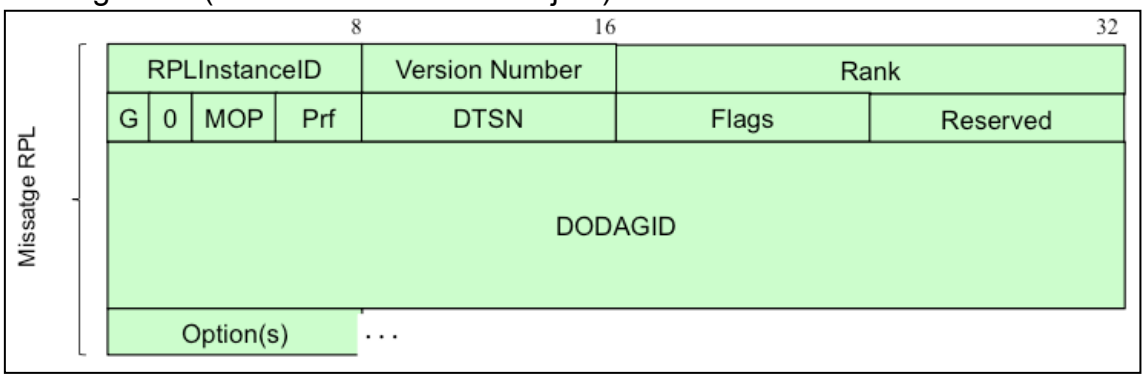


Fig. 35 Missatges DIO

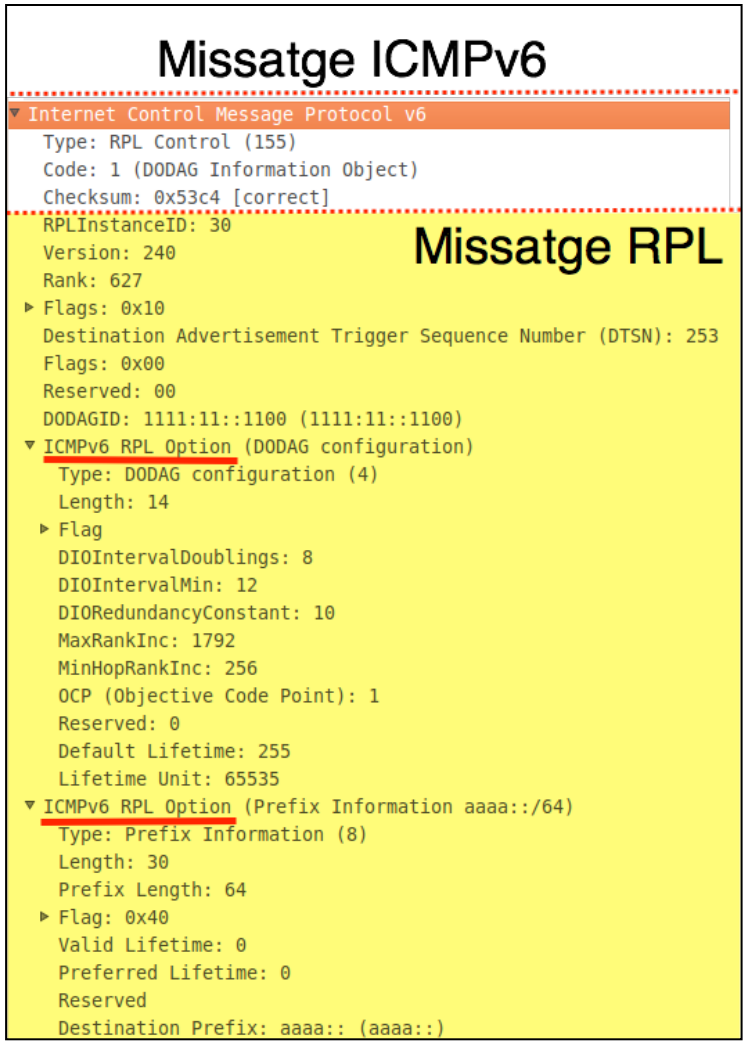


Fig. 36 Format d'un missatge ICMPv6

1.3. CoAP

Format missatges CoAP:

- Ver: versió
- T: tipus (CON,NON, ACK, RST)
- Option count: nombre d'opcions incloses en el missatge
- Code: petició/resposta i tipus
- Message ID: número de seqüència

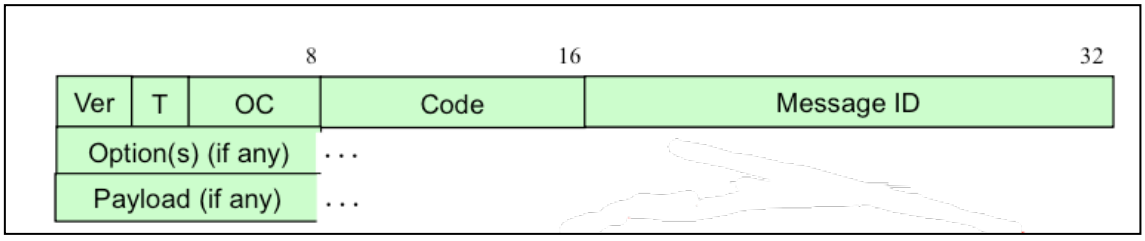


Fig. 37 Format del missatge CoAP

Exemple de missatge del tipus CON

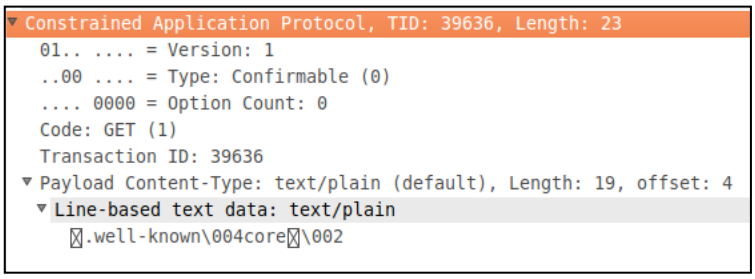


Fig. 38 Missatge tipus CON

Exemple de missatge del tipus ACK

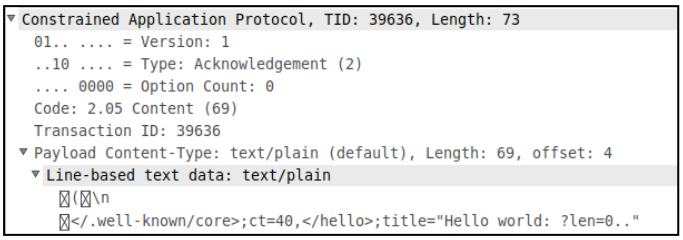


Fig. 39 Missatge ACK (CoAP)

Codis de petició CoAP

Code	Name	Reference
1	GET	[RFCXXXX]
2	POST	[RFCXXXX]
3	PUT	[RFCXXXX]
4	DELETE	[RFCXXXX]

Fig. 40 Codis de petició CoAP

- El mètode GET recupera una representació de la informació que correspon actualment al recurs identificat per la sol·licitud URI. Si la sol·licitud inclou una opció Acceptar, que indica el contingut en format preferit d'una resposta.

- El mètode POST sol·licita que la representació adjunta a la sol·licitud sigui processada. La funció real realitzat pel mètode POST es determina pel servidor d'origen i depenent del recurs de destinació. En general resulta en un nou recurs que s'està creant o el recurs de destinació en procés d'actualització.

- El mètode PUT sol·licita que el recurs identificat per la sol·licitud URI s'actualitza o es crea amb la representació adjunta. El format de representació és especificat pel tipus de material i contingut codificació en l'Opció Content-Format, si es proporciona.

- El mètode DELETE sol·licita que se suprimeixi el recurs identificat per la sol·licitud URI.

Codis de resposta CoAP

Code	Description	Reference
65	2.01 Created	[RFCXXXX]
66	2.02 Deleted	[RFCXXXX]
67	2.03 Valid	[RFCXXXX]
68	2.04 Changed	[RFCXXXX]
69	2.05 Content	[RFCXXXX]
128	4.00 Bad Request	[RFCXXXX]
129	4.01 Unauthorized	[RFCXXXX]
130	4.02 Bad Option	[RFCXXXX]
131	4.03 Forbidden	[RFCXXXX]
132	4.04 Not Found	[RFCXXXX]
133	4.05 Method Not Allowed	[RFCXXXX]
134	4.06 Not Acceptable	[RFCXXXX]
140	4.12 Precondition Failed	[RFCXXXX]
141	4.13 Request Entity Too Large	[RFCXXXX]
143	4.15 Unsupported Content-Format	[RFCXXXX]
160	5.00 Internal Server Error	[RFCXXXX]
161	5.01 Not Implemented	[RFCXXXX]
162	5.02 Bad Gateway	[RFCXXXX]
163	5.03 Service Unavailable	[RFCXXXX]
164	5.04 Gateway Timeout	[RFCXXXX]
165	5.05 Proxying Not Supported	[RFCXXXX]

Fig. 41 Tipus de Respostes CoAP

Exemple Observe Option (CoAP)

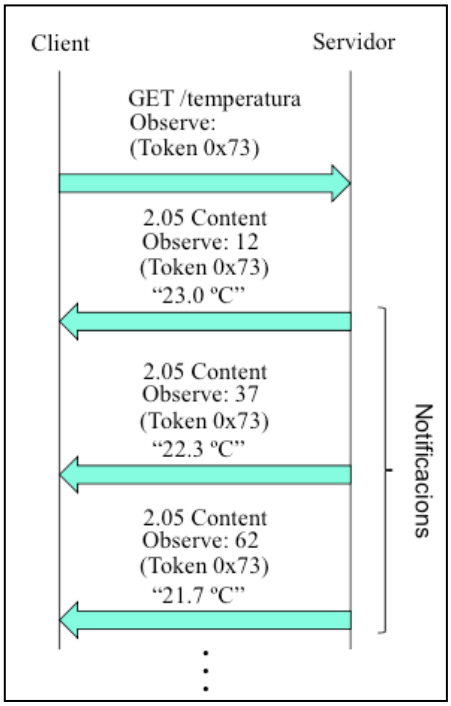


Fig. 42 Observe Option (CoAP)

Exemple Block Option (CoAP)

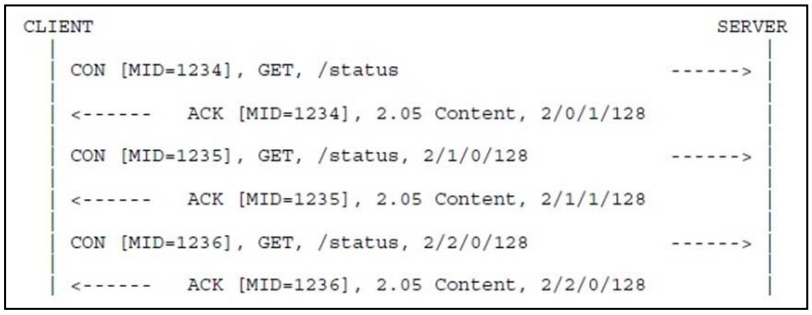


Fig. 43 Block Option (CoAP)

ANNEX B: HARDWARE DE L'ECONOTAG

Interfície Radio MC1322x

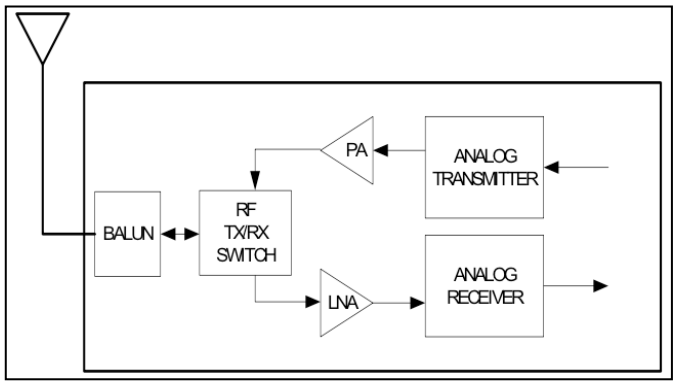


Fig. 44 Ràdio MC1322x

Diagrama de Blocs simplificat del MC1322x

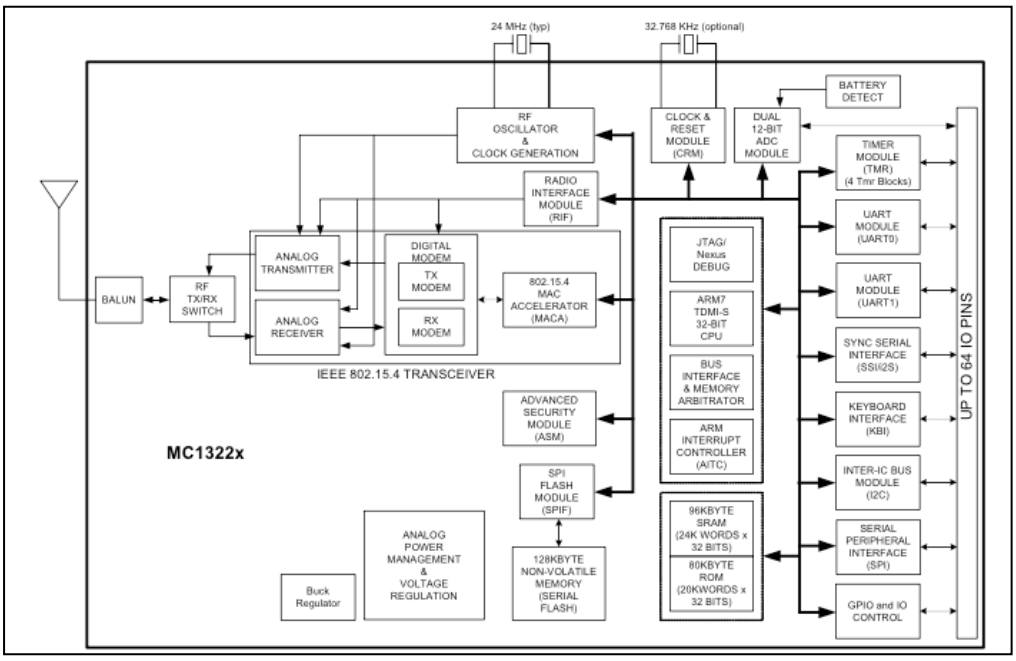


Fig. 45 Diagrama de blocs MC1322x

Característiques del mòdul MACA:

- redueix la càrrega de CPU
- generador de nombres aleatoris de 32 bits
- "*warm-up*" recepció (rx) : 72 µs
- "*warm-up*" transmissió (tx): 92 µs

Memòria MC1322x:

- RAM: 96 Kbytes
- ROM: 80 Kbytes
- Serial FLASH: 128 Kbytes

ANNEX C: INSTAL·LACIÓ DE L'ENTORN

La primera guia de com partir i poder fer servir els nostres dispositius de hardware, les motes Econotag; serà la instal·lació de l'entorn. En primer lloc ens hem baixat el Sistema Operatiu Contiki de la seva plana web principal que és aquesta: http://www.contiki-os.org/

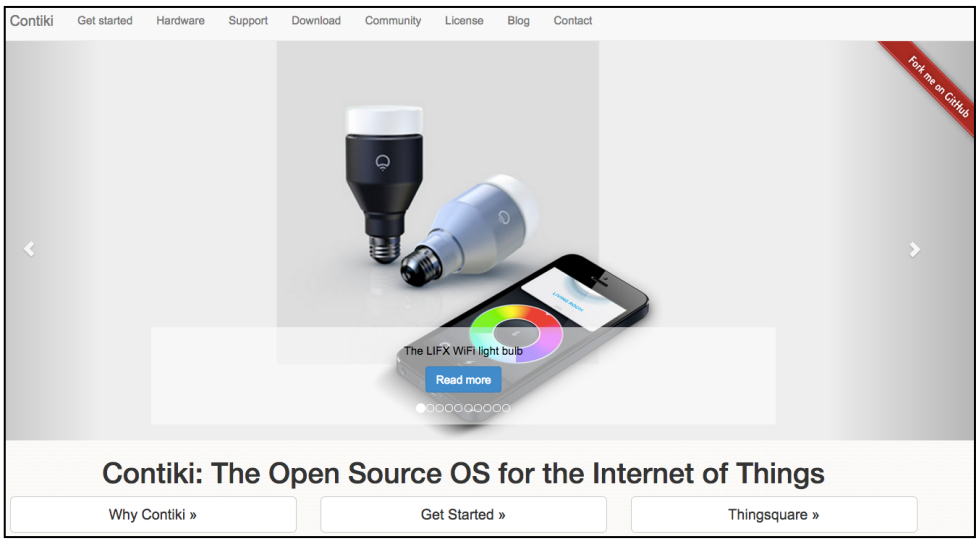


Fig. 46 Pàgina principal Contiki

Podem veure en la imatge de dalt, quin aspecte té la pàgina web del sistema operatiu, un cop a dins, ens ficarem a la pestanya que diu "*Get Started*".
Allà disposarem de totes les instruccions necessàries per tal de procedir amb la instal·lació de tot l'entorn.
La instal·lació consta dels següents passos:

- En primer lloc ens baixarem la imatge d'instant Contiki. En aquest cas, jo he optat per la imatge més recent del Sistema Operatiu, la versió Instant Contiki 2.7. Podem obtenir-lo accedint al següent enllaç:
  http://sourceforge.net/projects/contiki/files/Instant%20Contiki/

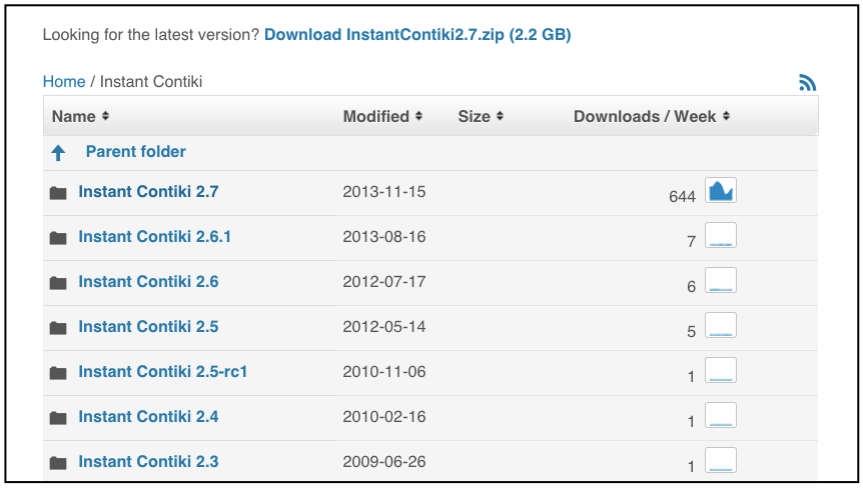


Fig. 47 Pàgina de descàrrega d'Instant Contiki

Instant Contiki és tot un entorn de desenvolupament de Contiki en una sola descàrrega. Es tracta d'una màquina virtual Ubuntu Linux que s'executa amb VMware i que té Contiki amb totes les eines de desenvolupament, compiladors i simuladors utilitzats en el desenvolupament instal·lades.

Un cop ens hem baixat la imatge d'Instant Contiki, i la descomprimim, podrem comprovar quin és el contingut del paquet:

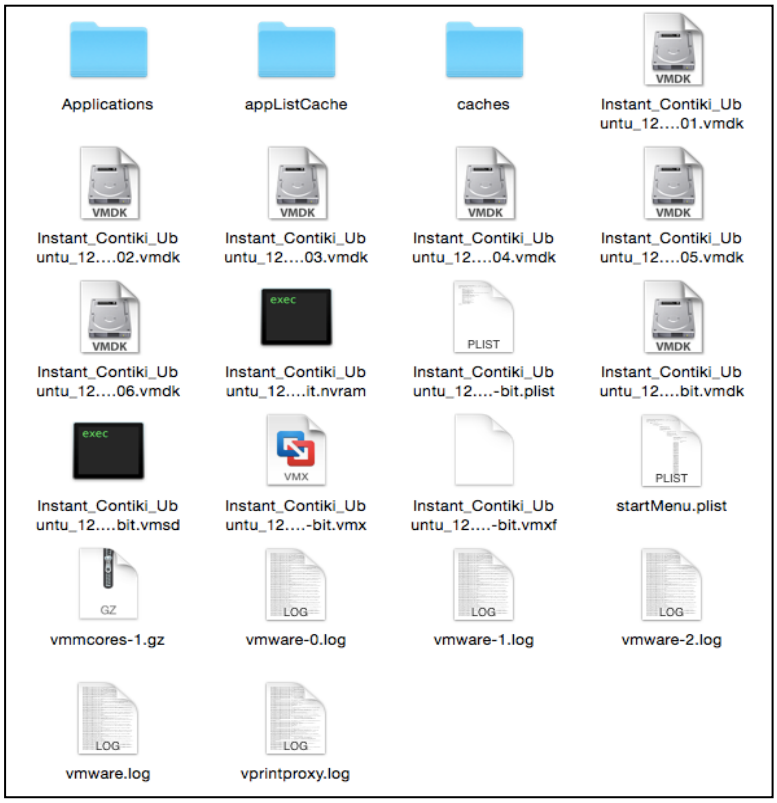


Fig. 48 Contingut Paquet Instant Contiki 2.7

- A continuació, un cop ens hem baixat i descomprimit la imatge anterior, el següent pas serà el d'obtenir l'entorn de virtualització VMware, ens el descarregarem i procedirem amb la instal·lació.
  Cal dir que l'VMware Player és de lliure distribució tant per Windows com per Linux; això sí, la última versió, la 7.0, només és compatible amb dispositius de 64 bits. Per tant en els ordinadors del laboratori on farem el desplegament dels escenaris i les futures pràctiques per a la Universitat, el laboratori C4-331G, allà s'optarà per fer la instal·lació de la versió 6 de VMware Player.

Fig. 49 Descàrrega VMWare

S'opta per utilitzar l'entorn de virtualització VMware degut a què en aquest cas per exemple, fer servir un altre possible entorn com pot ser VirtualBox no és recomanable ja que ha donat alguns problemes a l'hora de reconèixer els ports USB. Si de totes maneres algú preferís fer servir VirtualBox enlloc de VMware, es poden mirar de resoldre aquests problemes seguint els passos que s'expliquen en aquest enllaç:

http://elrincondelgnulinux.blogspot.com.es/2013/02/ubuntu-como-utilizar-dispositivos-usb.html

- A continuació d'haver instal·lat l'entorn de virtualització, el següent pas serà el d'executar la imatge d'instant Contiki, és el fitxer .vmx del directori que ens hem descarregat.

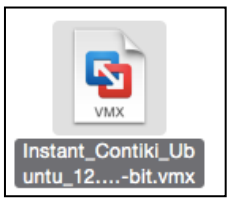


Fig. 50 Instant Contiki vmx

Ens arrancarà la màquina virtual amb Ubuntu, i seleccionarem l'usuari Instant Contiki, amb *password*: user

Un cop fet això, ja tindrem Instant Contiki Funcionant.

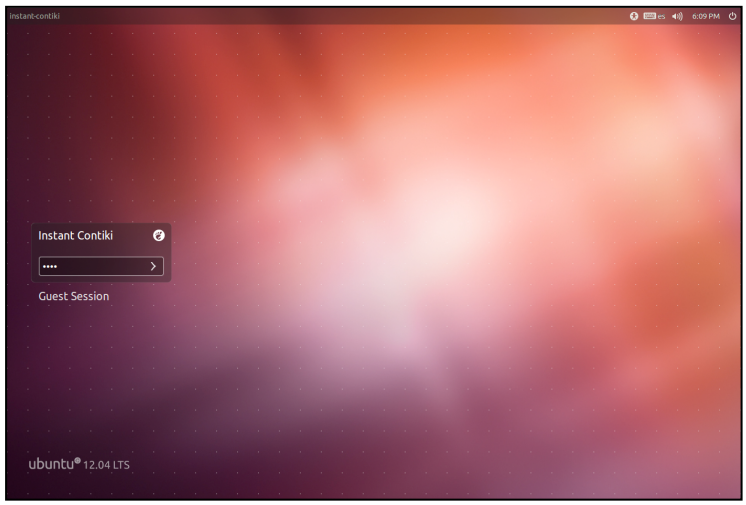


Fig. 51 Pantalla inici sessió Instant Contiki

ANNEX D: ESTRUCTURA DE CARPETES CONTIKI O.S.

A dintre de la carpeta home tinc ubicades les carpetes contiki, així com la versió corresponent a contiki-2.7, i també per últim la carpeta contiki-git.

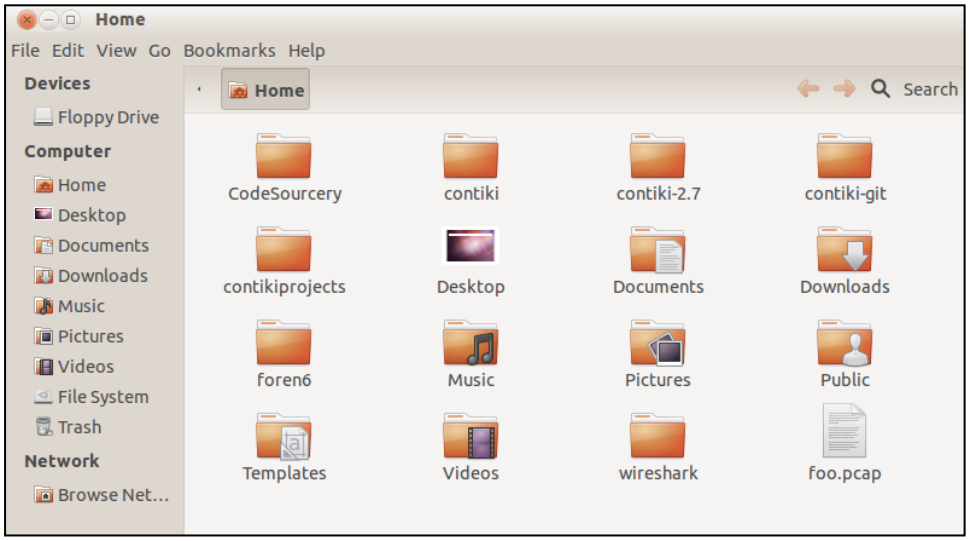


Fig. 52 Estructura carpeta home d'usuari

Per tant tenim a /home/user:

/contiki
/contiki-2.7
/contiki-git

L'estructura de carpetes seguida a dintre de /home/user/contiki-2.7/cpu:

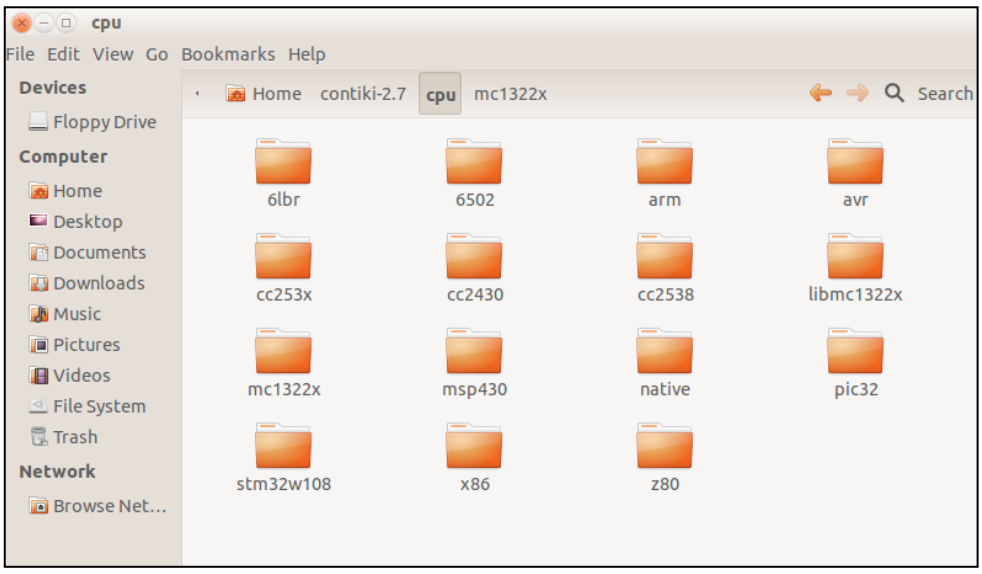


Fig. 53 Estructura carpeta contiki-2.7

Incorpora les llibreries "*6lbr*", "*libmc1322x*", i també "*mc1322x*".

ANNEX E: EINES I LLIBRERIES DE CONTIKI

4.1. Llibreria libmc1322x

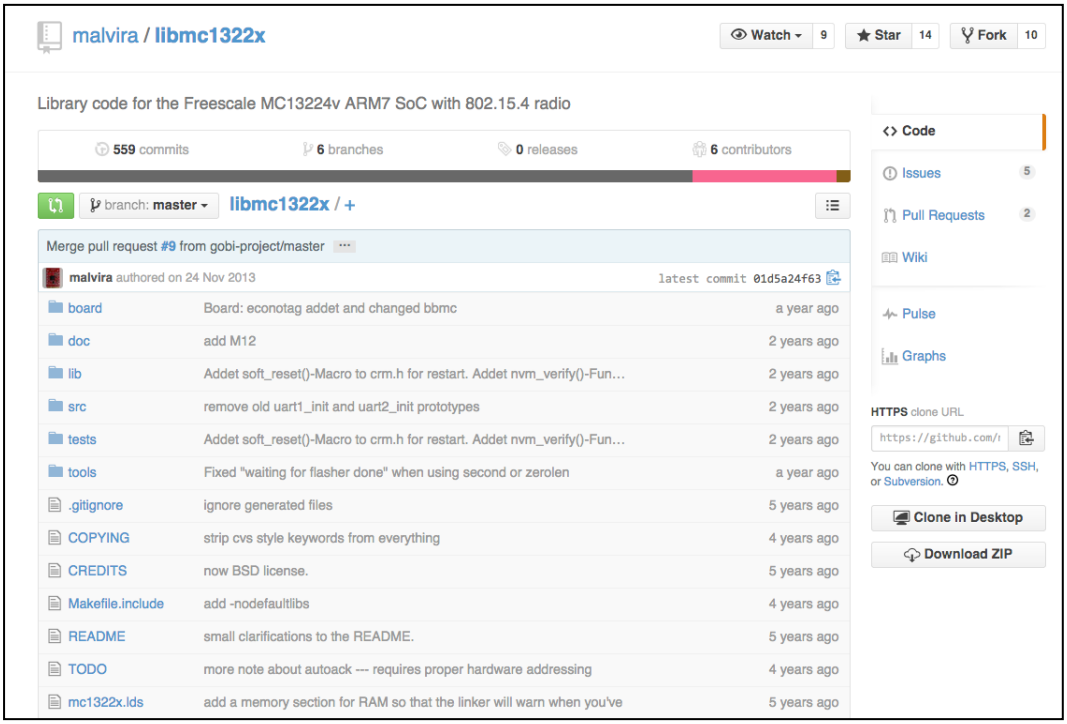


Fig. 54 Repositori de Github: llibreria libmc1322x

4.2. Repositori 6LBR de Github

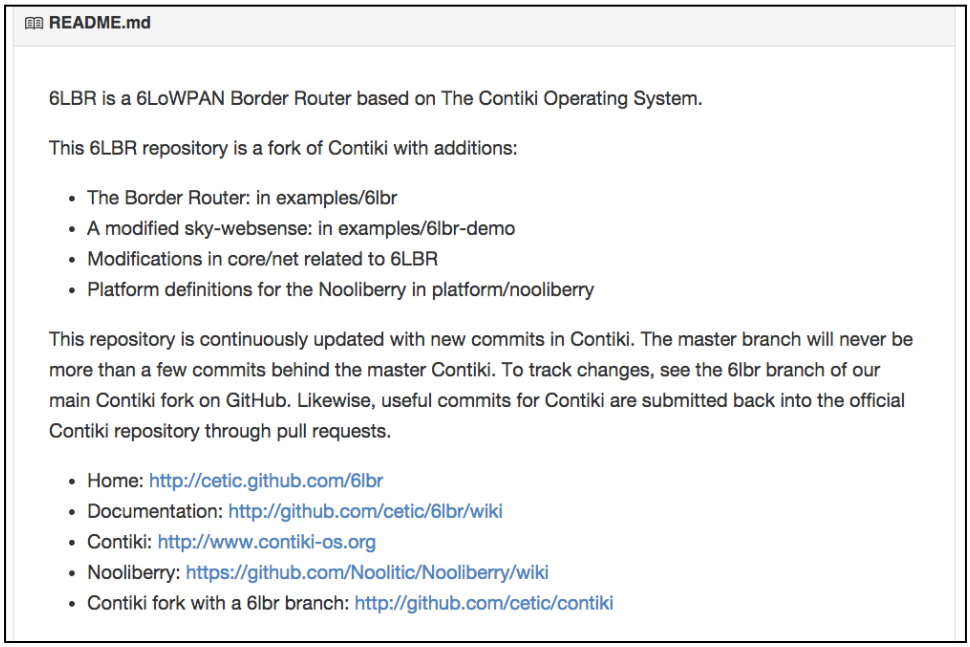


Fig. 55 6LBR Github

4.3. Tunslip6

Captura de pantalla tunslip6:

```
user@instant-contiki:~/contiki-2.7/cpu/6lbr/tools$ sudo ./tunslip6 -s /dev/ttyUS
B1 aaaa::1/64
********SLIP started on ``/dev/ttyUSB1''
opened tun device ``/dev/tun0''
ifconfig tun0 inet `hostname` up
ifconfig tun0 add aaaa::1/64
ifconfig tun0 add fe80::0:0:0:1/64
ifconfig tun0

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00
          inet addr:127.0.1.1  P-t-P:127.0.1.1  Mask:255.255.255.255
          inet6 addr: fe80::1/64 Scope:Link
          inet6 addr: aaaa::1/64 Scope:Global
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
```

Fig. 56 Tunslip

Ping amb tunslip:

```
user@instant-contiki:~$ ping6 aaaa::1
PING aaaa::1(aaaa::1) 56 data bytes
64 bytes from aaaa::1: icmp_seq=1 ttl=64 time=0.032 ms
64 bytes from aaaa::1: icmp_seq=2 ttl=64 time=0.042 ms
64 bytes from aaaa::1: icmp_seq=3 ttl=64 time=0.077 ms
64 bytes from aaaa::1: icmp_seq=4 ttl=64 time=0.049 ms
64 bytes from aaaa::1: icmp_seq=5 ttl=64 time=0.036 ms
```

Fig. 57 Ping amb tunslip

4.4. Connexió dels Econotags a l'USB de la màquina virtual

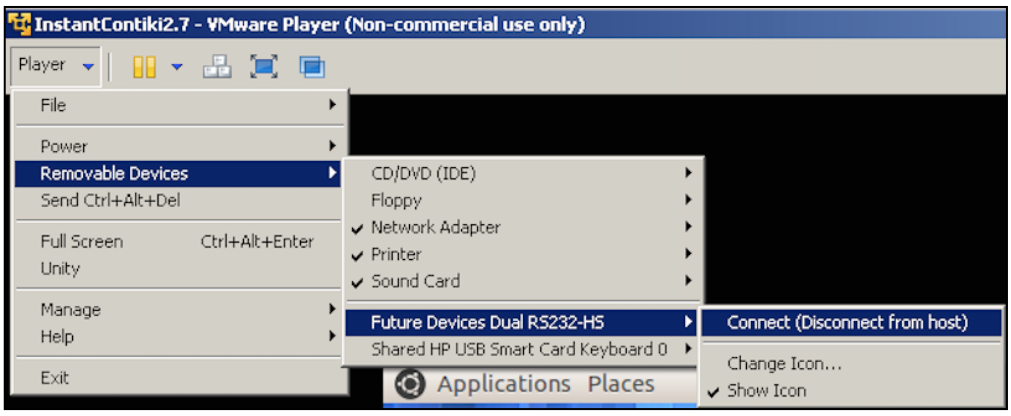


Fig. 58 Connexió plaques Econotag al port USB de VMware

ANNEX F: CAPTURES DE PANTALLA D'EXEMPLES

Aquest annex recull les captures de pantalla, i també alguns programes de Contiki d'exemples amb els que he anat fent proves de carregar-los als dispositius Econotags.
Aquests programes en alguns casos fan ús de les eines que hem estat veient amb anterioritat, també poden formar part d'alguna llibreria de contiki, o simplement son 'tests' inclosos entre els fitxers d'exemples que ens dóna contiki.
Aquests exemples m'han servit per tant per tenir una base sobre la qual un cop he recollit informació i investigat al respecte, després s'han muntat els escenaris.

5.1. Exemple Hello World (Càrrega a la RAM)

```
.......................CON♦CONNECT
Size: 30344 bytes
Sending /home/user/contiki-2.7/examples/hello-world/hello-world_econotag.bin
done sending files.
mc1322x init
vbatt: 3250 mV
NVM failed without buck, trying with buck
buck ok
starting vbatt monitor
config bad magic ffff
config bad version ffff
flash invalid
mc1322x config:
  magic:    1322
  version:  1
  eui:      0000000000000000
  channel:  15
  power:    17
  flags:    fffffc5
     demod:    1
     autoack:  0
     nvm type: 1
trying to start 32kHz xtal
32kHz xtal started
trim xtal for M12
setting panid 0xcdab
setting short mac 0xffff
setting long mac 0x00050c2a_8c6e1be4
Rime configured with address 00:05:0C:2A:8C:6E:1B:E4
Starting 'Hello world process'
Hello, world
```

Fig. 59 Consola Hello World

5.2. Sobre com carregar i esborrar de la Flash

```
user@instant-contiki:~$ cd /home/user/contiki-2.7/cpu/mc1322x/tools
user@instant-contiki:~/contiki-2.7/cpu/mc1322x/tools$ sudo ./mc1322x-load.pl -f
/home/user/contiki-2.7/cpu/libmc1322x/tests/flasher_m12.bin -s /home/user/contik
i-2.7/examples/hello-world/hello-world_econotag.bin -t /dev/ttyUSB1
[sudo] password for user:
..CONNECT
Size: 11416 bytes
Sending /home/user/contiki-2.7/cpu/libmc1322x/tests/flasher_m12.bin
secondary send...
.Detecting internal nvm
nvm_detect returned: 0x00 type is: 0x00000001
nvm_erase returned: 0x00
ready
Size: 30344 bytes
Sending /home/user/contiki-2.7/examples/hello-world/hello-world_econotag.bin
done sending files.
write_len: 0x00007688
write successfully done
prog_len: 0xea0001ff
flasher done
```

Fig. 60 Exemple càrrega a la Flash

Un exemple de la crida anterior del fitxer flasher_m12.bin

sudo ./mc1322x-load.pl -f /home/user/contiki-2.7/cpu/libmc1322x/tests/flasher_m12.bin -s /home/user/contiki-2.7/examples/hello-world/hello-world_econotag.bin -t /dev/ttyUSB1

***Al respecte sobre flashejar l'Econotag, cal tenir en compte:**

(I): Per carregar imatges a l'Econotag, el flasher adequat és el *flasher_m12.bin.* Per comprovar si s'ha fet el flash correctament hem d'estar atents a la sortida que obtenim, ja que si veiem un warning/error, és que no s'ha fet correctament.

(II): Quan carreguem un programa, ens apareix a la consola un "done sending files" s'ha de prémer novament el botó 'reset' de l'Econotag i veurem com executa el programa amb què l'hem flashejat.

Esborrar un fitxer de la memòria flash:

```
user@instant-contiki:~/contiki-git$ cd cpu/mc1322x/tools/ftditools/
user@instant-contiki:~/contiki-git/cpu/mc1322x/tools/ftditools$ sudo ./bbmc -l r
edbee-econotag erase
[sudo] password for user:
Found 2 devices with vendor id 0x0403 product id 0x6010

  [0]   Manufacturer: FTDI, Description: Dual RS232-HS, Serial ?
  [1]   Manufacturer: FTDI, Description: Dual RS232-HS, Serial ?

Use which device? 1
Opening device 1 interface 1 using layout redbee-econotag
setting VREF2 erase
toggle reset
waiting for erase
setting VREF2 normal
toggle reset
done.
user@instant-contiki:~/contiki-git/cpu/mc1322x/tools/ftditools$
```

Fig. 61 Esborrar Econotag

Ens apareix una opció que ens detecta els Econotags que tenim connectats a la màquina virtual amb Contiki, i ens deixa triar quin volem esborrar.

5.3. Exemples d'Emissor i Receptor (UDP-RPL)

5.3.1. Exemple Dispositiu com a Transmissor Broadcast UDP

A dintre de les carpetes del sistema operatiu contiki-2.7 ens trobem que hi ha exemples que fan servir udp-rpl, per tal de transmetre o rebre paquets. Un d'aquests exemples és el transmissor broadcast UDP que el trobem en el següent directori del nostre Sistema Operatiu:

/home/user/contiki-2.7/examples/ipv6/simple-udp-rpl

Una vegada ens hem ubicat al directori de l'exemple, podrem crear binari fent:

make TARGET=econotag broadcast-example

Després procedirem a enviar l'arxiu binari a l'Econotag amb:

cd /home/user/contiki-2.7/cpu/mc1322x/tools

sudo ./mc1322x-load.pl -f /home/user/contiki-2.7/examples/ipv6/simple-udp-rpl/broadcast-example_econotag.bin -t /dev/ttyUSB1

```
mc1322x config:
  magic:    1322
  version:  1
  eui:      00050c2a8c0bce83
  channel:  15
  power:    17
  flags:    ffffffc5
     demod:    1
     autoack:  0
     nvm type: 1
trying to start 32kHz xtal
32kHz xtal started
trim xtal for M12
setting panid 0xcdab
setting short mac 0xffff
setting long mac 0x00050c2a_8c0bce83
Rime configured with address 00:05:0C:2A:8C:0B:CE:83
nullmac nullrdc, channel check rate 100 Hz, radio channel 26
Tentative link-local IPv6 address fe80:0000:0000:0000:0205:0c2a:8c0b:ce83
Starting 'UDP broadcast example process'
Sending broadcast
Sending broadcast
Sending broadcast
```

Fig. 62 Exemple Broadcast

Aquí podem veure el format del paquet UDP Broadcast (destinació ff02::1):

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::205:c2a:8c34:b1c7	ff02::1	UDP	29	Source port: se

```
▸ Frame 2: 29 bytes on wire (232 bits), 29 bytes captured (232 bits) on interface 0
▾ IEEE 802.15.4 Data, Dst: Broadcast, Src: NetworkP_2a:8c:34:b1:c7
  ▸ Frame Control Field: Data (0xc841)
    Sequence Number: 107
    Destination PAN: 0xabcd
    Destination: 0xffff
    Extended Source: NetworkP_2a:8c:34:b1:c7 (00:05:0c:2a:8c:34:b1:c7)
▾ 6LoWPAN
  ▸ IPHC Header
    Source: fe80::205:c2a:8c34:b1c7 (fe80::205:c2a:8c34:b1c7)
    Destination: ff02::1 (ff02::1)
  ▸ UDP header compression
    Source port: 1234
    Destination port: 1234
    UDP checksum: 0xe4a8
▾ Internet Protocol Version 6, Src: fe80::205:c2a:8c34:b1c7 (fe80::205:c2a:8c34:b1c7), Dst: ff02::1 (ff02::1)
    0110 .... = Version: 6
  ▸ .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
    .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
    Payload length: 12
    Next header: UDP (17)
    Hop limit: 64
    Source: fe80::205:c2a:8c34:b1c7 (fe80::205:c2a:8c34:b1c7)
    Destination: ff02::1 (ff02::1)
```

Fig. 63 Paquet Broadcast UDP

5.3.2. Dispositiu com a Emissor Unicast UDP

De la mateixa manera que el programa anterior que hem carregat al nostre Econotag, el qual es tractava d'un exemple d'un "sender" broadcast UDP, també tenim l'exemple ubicat a la mateixa carpeta que abans:
2.7/examples/ipv6/simple-udp-rpl
Seguint els mateixos passos que abans el podem carregar: compilem el programa per al nostre target Econotag, un cop tenim el binari procedim a cridar l'mc1322x-load i el carreguem en un dels ports USB.

```
mc1322x config:
  magic:    1322
  version:  1
  eui:      00050c2a8c0bce83
  channel:  15
  power:    17
  flags:    fffffffc5
     demod:    1
     autoack:  0
     nvm type: 1
trying to start 32kHz xtal
32kHz xtal started
trim xtal for M12
setting panid 0xcdab
setting short mac 0xffff
setting long mac 0x00050c2a_8c0bce83
Rime configured with address 00:05:0C:2A:8C:0B:CE:83
nullmac nullrdc, channel check rate 100 Hz, radio channel 26
Tentative link-local IPv6 address fe80:0000:0000:0000:0205:0c2a:8c0b:ce83
Starting 'Unicast sender example process'
IPv6 addresses: aaaa::205:c2a:8c0b:ce83
fe80::205:c2a:8c0b:ce83
```

Fig. 64 Unicast Sender

5.3.3. Dispositiu com a Receptor

A banda de poder fer servir el dispositiu Econotag com a emissor de paquets UDP, a la mateixa carpeta d'abans 'simple-udp-rpl', tenim allà un exemple que fa servir el dispositiu com a receptor. El fitxer c de l'exemple es troba en el següent path del sistema:

*contiki/examples/ipv6/simple-udp-rpl/**unicast-receiver.c***

De la mateixa manera en que ho hem fet abans, haurem de ubicar-nos a la carpeta de l'exemple i compilar els binaris:

cd /home/user/contiki-2.7/examples/ipv6/simple-udp-rpl
make TARGET=econotag unicast-receiver

Per a carregar l'exemple a la placa farem el següent:

cd /home/user/contiki-2.7/cpu/mc1322x/tools
sudo ./mc1322x-load.pl -f /home/user/contiki-2.7/examples/ipv6/simple-udp-rpl/unicast-receiver_econotag.bin -t /dev/ttyUSB1

```
trying to start 32kHz xtal
32kHz xtal started
trim xtal for M12
setting panid 0xcdab
setting short mac 0xffff
setting long mac 0x00050c2a_8c14d607
Rime configured with address 00:05:0C:2A:8C:14:D6:07
nullmac nullrdc, channel check rate 100 Hz, radio channel 26
Tentative link-local IPv6 address fe80:0000:0000:0000:0205:0c2a:8c14:d607
Starting 'Unicast receiver example process'
IPv6 addresses: aaaa::205:c2a:8c14:d607
fe80::205:c2a:8c14:d607
Data received from fe80::205:c2a:8c34:b1c7 on port 1234 from port 1234 with leng
th 4: 'Test'
Data received from fe80::205:c2a:8c34:b1c7 on port 1234 from port 1234 with leng
th 4: 'Test'
Data received from fe80::205:c2a:8c34:b1c7 on port 1234 from port 1234 with leng
th 4: 'Test'
```

Fig. 65 Unicast Receiver

Més endavant en un escenari on hi intervenen un emissor i un receptor, podrem comprovar com els paquets arriben al terminal del dispositiu que fa de receptor.

5.3.4. Escenari amb un emissor i un receptor

Vistos ja per separat els diferents programes que ens permeten fer servir els nostres dispositius Econotags com a emissor i receptor respectivament. El següent pas lògic és fer una prova on intervinguin dos dispositius Econotag alhora i que un faixi de node emissor i l'altre faci de node receptor.
En la captura de baix podem veure el resultat en consola d'un escenari que inclou un dispositiu Econotag com a broadcast sender, i l'altra fent d'unicast receiver.

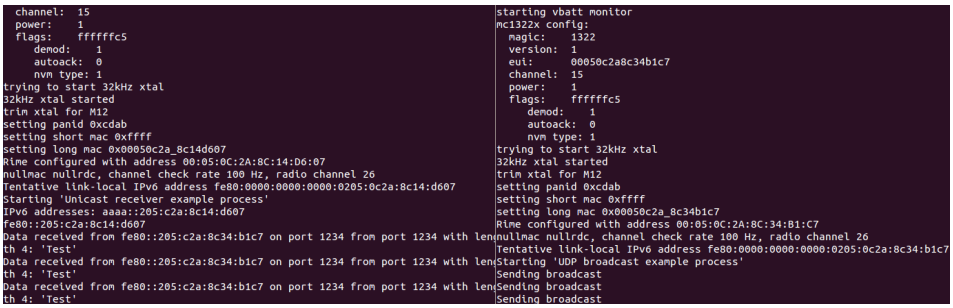


Fig. 4.66 Emissor + Receptor

Veiem en el terminal del receptor com es decodifiquen els paquets, si els analitzem amb un sniffer, podem veure'ls amb Wireshark; el camp de dades podem veure com té longitud 4, i a baix podem veure la paraula 'Test'.

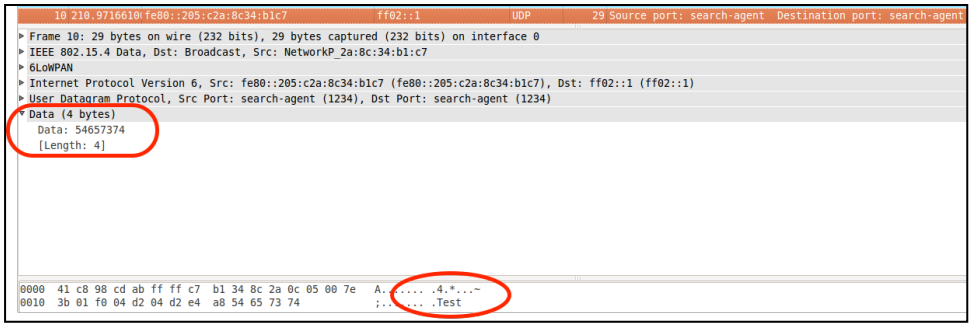


Fig. 4.67 Paquet 'Test' UDP

*De cara a les proves de laboratori, o a les sessions pràctiques que es facin en assignatures: Una configuració interessant seria també la d'habilitar una xarxa amb un emissor broadcast amb múltiples dispositius receptors descodificant els paquets.

5.4. Econotag com a Sniffer

Arxius binari que es crida per capturar paquets:

Fig. 68 Arxiu binari del 'rftest-rx'

Càrrega del programa rftest-rx vist en el terminal

```
...CONNECT
Size: 17736 bytes
Sending /home/user/Desktop/libmc1322x-master/tests/rftest-rx_m12.bin
done sending files.
mc1322x-test: rftest-rx
board: m12
```

Fig. 69 Programa 'rftest-rx' (per fer d'Sniffer)

Si tenim un altre dispositiu que envia paquets en el nostre mateix canal, els podrem veure (això és abans de llençar l'script per capturar en temps real amb Wireshark, és del propi rftest-rx):

```
rftest-rx --- len 0x19 lqi 0x8d rx_time 0x00e3c6fc
41 c8 bf cd ab ff ff 52 dd 63 8c 2a 0c 05 00 7a
3b 3a 1a 9b 00 ef 3b 00 00
```

Fig. 70 Paquets rftestrx (abans script)

Captura en "Temps Real":
Després d'haver llençat l'script per fer la captura en temps real. Podrem veure en el terminal on executem l'script com es descodifiquen els paquets:

```
new packet: 1436456754 576258 25 25
dataline: 41 c8 c5 cd ab ff ff 52 dd 63 8c 2a 0c 05 00 7a
dataline: 3b 3a 1a 9b 00 ef 3b 00 00
rftestline: 1436456814 441404 25 rftest-rx --- len 0x19 lqi 0x8d rx_time 0x04754
af3
```

Fig. 71 Paquets *rftestrx2pcap* (després script)

Captura en "Diferit":
Per bolcar els paquets en un fitxer pcap, que consisteix en cridar l'script '*rftestrx2pcap.py*', indicant en quin port USB estem capturant, a continuació el canal, i per últim el fitxer on volem guardar la captura.
Un exemple de com es fa seria aquest:

sudo ./Desktop/libmc1322x-master/tools/rftestrx2pcap.py /dev/ttyUSB1 12 /Desktop/fifo.pcap

5.4.1. Exemple 'rftest-tx' (transmissor) i 'rftest-rx' (receptor)

Per a configurar el dispositiu Econotag per a funcionar de sniffer, com ja hem vist abans, fem servir el programa 'rftest-rx' que trobem en el directori '*tests*' de la llibreria libmc1322x. La primera prova de captura de paquets, l'he fet utilitzant un programa d'exemple de transmissor, que trobem en la mateixa carpeta de testos de la llibreria; el programa s'anomena 'rftest-tx'.
He fet servir aquest programa primer per provar el tema de l'sniffer, ja que els dos programes estan pensats per a funcionar en el mateix canal ràdio, de tal forma que ja es veuran directament els dos dispositius i no hi haurà cap problema a l'hora de veure els paquets entre ells.
Si tenim dos Econotags, i carreguem els dos programes, podrem veure com quan arranquem en el terminal l'execució d'aquests, es poden veure els paquets que envien els dos programes. A l'esquerra tenim l'emissor 'rftest-tx', a la dreta el receptor 'rftest-rx'.

```
len 0x10 lqi 0x00 rx_time 0x00000000
48 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f

len 0x10 lqi 0x00 rx_time 0x00000000
58 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f

len 0x10 lqi 0x00 rx_time 0x00000000
68 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
```

```
len 0x10 lqi 0xdb rx_time 0x00554b5e
48 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f

len 0x10 lqi 0xdb rx_time 0x00559606
58 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f

len 0x10 lqi 0xdb rx_time 0x0055e0ad
68 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
```

Fig. 72 Terminal: paquets rftest-tx / rx

Podem veure en la captura anterior com els paquets tenen els següents camps:

- len: longitud del paquet
- lqi: link quality indication
- rx_time: temps de recepció

En aquests dos programes apareixen també alguns dels paràmetres de configuració que fan referència a la capa física, així com les funcions respectives que els defineixen. En el capítol dedicat a l'anàlisi de les configuracions s'analitza el que fa referència a la configuració dels canals ràdio i també a la potència de transmissió.

A continuació hem executat l'script per a fer la captura en temps real amb Wireshark, dels dos programes anteriors; per una banda tenim el transmissor 'rftest-tx' i per l'altra el receptor o sniffer.

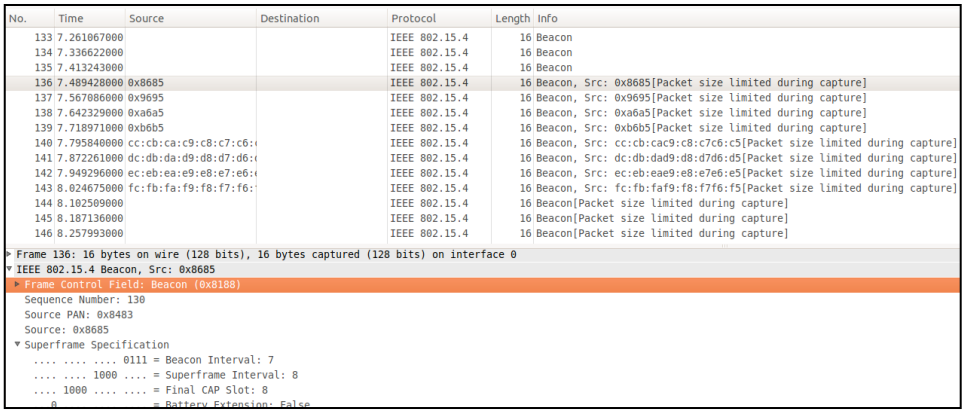


Fig. 73 Captura rftest-tx / rftest-rx

ANNEX G: CAPTURES DELS ESCENARIS FINALS

6.1. Càrrega dels nodes de l'escenari

Consola de càrrega del Border Router:

```
mc1322x init
vbatt: 3283 mV
NVM failed without buck, trying with buck
buck ok
starting vbatt monitor
mc1322x config:
  magic:    1322
  version:  1
  eui:      00050c2a8c1ade2b
  channel:  12
  power:    8
  flags:    fffffc5
     demod:    1
     autoack:  0
     nvm type: 1
trying to start 32kHz xtal
32kHz xtal started
trim xtal for M12
setting panid 0xcdab
setting short mac 0xffff
setting long mac 0x00050c2a_8c1ade2b
Rime configured with address 00:05:0C:2A:8C:1A:DE:2B
nullmac nullrdc, channel check rate 100 Hz, radio channel 11
Tentative link-local IPv6 address fe80:0000:0000:0000:0205:0c2a:8c1a:de2b
Starting 'Border router process' 'Web server'
◆?P◆◆?P◆◆?P◆◆?P◆◆?P◆◆?P◆◆?P◆◆?P◆◆?P◆◆?P◆◆?P◆^Cuser@instant-contiki:~/con
```

Fig. 74 Border Router

Quan feim tunslip al Border Router:

```
ifconfig tun0 add aaaa::1/64
ifconfig tun0 add fe80::0:0:0:1/64
ifconfig tun0

tun0      Link encap:UNSPEC  HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00
-00
          inet addr:127.0.1.1  P-t-P:127.0.1.1  Mask:255.255.255.255
          inet6 addr: fe80::1/64 Scope:Link
          inet6 addr: aaaa::1/64 Scope:Global
          UP POINTOPOINT RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

*** Address:aaaa::1 => aaaa:0000:0000:0000
Got configuration message of type P
Setting prefix aaaa::
Server IPv6 addresses:
 aaaa::205:c2a:8c1a:de2b
 fe80::205:c2a:8c1a:de2b
```

Fig. 75 Configuració adreces Tunslip6

Com poden veure ens assigna el rang aaaa::1/64

Càrrega Erbium Example Server:

```
mc1322x init
vbatt: 3254 mV
NVM failed without buck, trying with buck
buck ok
starting vbatt monitor
mc1322x config:
  magic:    1322
  version:  1
  eui:      00050c2a8c7f6c08
  channel:  12
  power:    8
  flags:    fffffffc5
     demod:    1
     autoack:  0
     nvm type: 1
trying to start 32kHz xtal
32kHz xtal started
trim xtal for M12
setting panid 0xcdab
setting short mac 0xffff
setting long mac 0x00050c2a_8c7f6c08
Rime configured with address 00:05:0C:2A:8C:7F:6C:08
nullmac nullrdc, channel check rate 100 Hz, radio channel 22
Tentative link-local IPv6 address fe80:0000:0000:0000:0205:0c2a:8c7f:6c08
Starting 'Erbium Example Server'
```

Fig. 76 Erbium Example Server

6.2. Recursos Web amb Firefox (Copper)

Fig. 54 Extensió Copper (Cu)

Pàgina web del Border Router: ContikiRPL

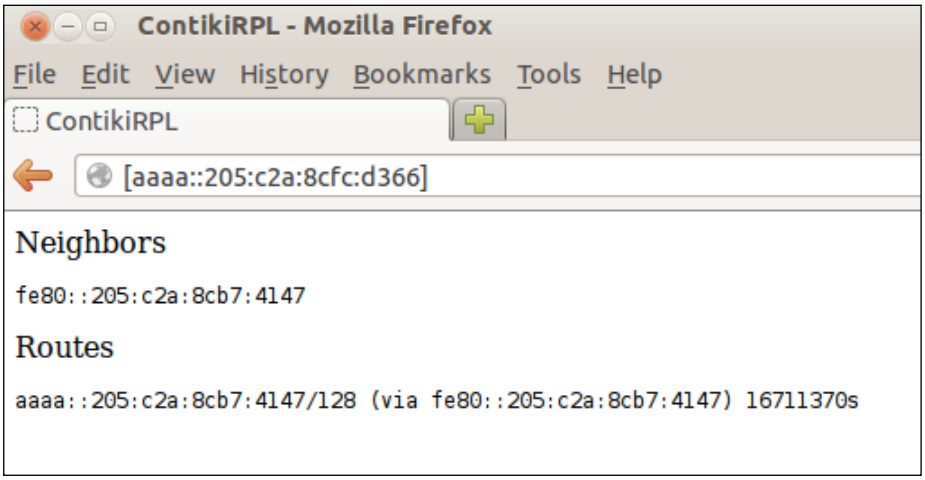


Fig. 77 Pàgina del Border Router

Pàgina web del Servidor de CoAP

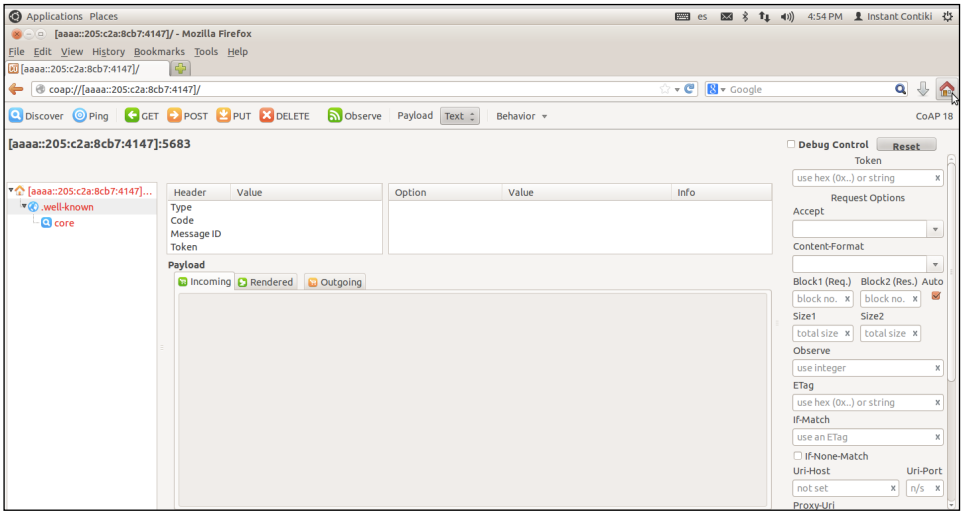


Fig. 78 Pàgina del servidor CoAP

6.2.1. Peticions REST amb Copper

- Opció Discover: Descobriment de Recursos. Ens permet veure els recursos disponibles del servidor de CoAP.

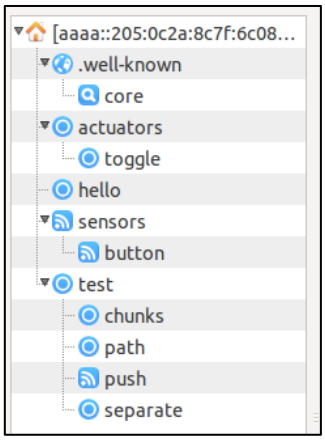


Fig. 79 Discover mota Econotag

- Petició tipus GET del recurs: *'.well-known / core'*

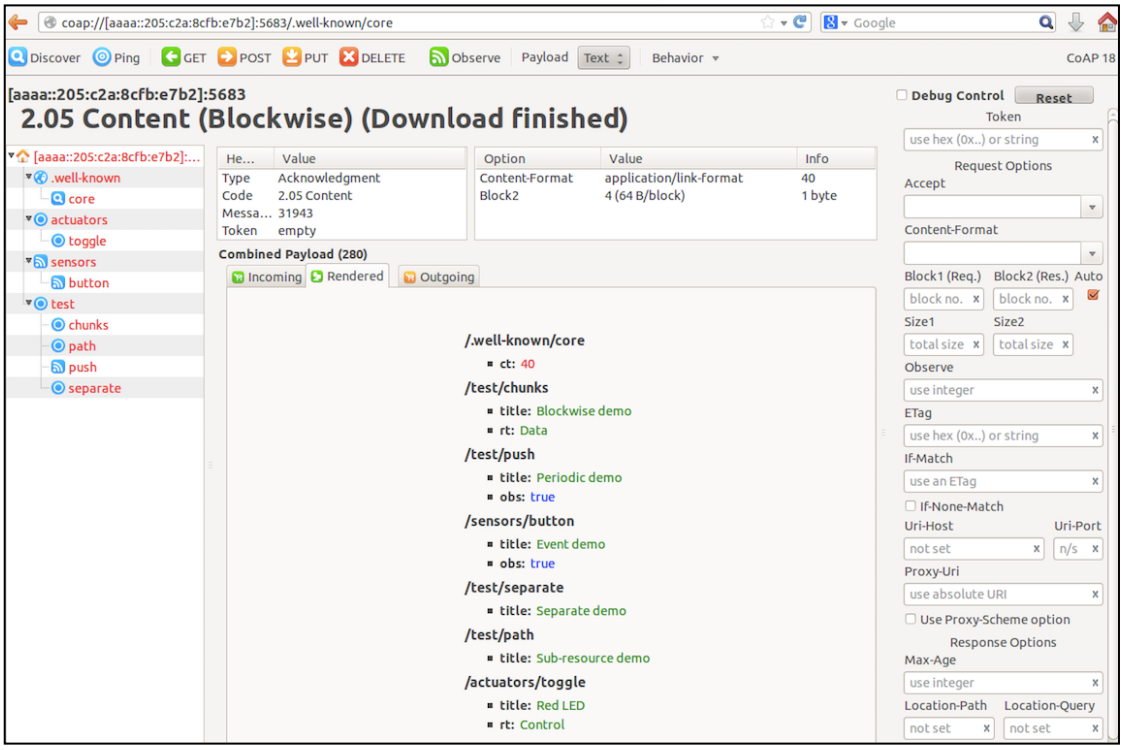


Fig. 80 Petició GET well-known/core

Format del paquet GET de CoAP

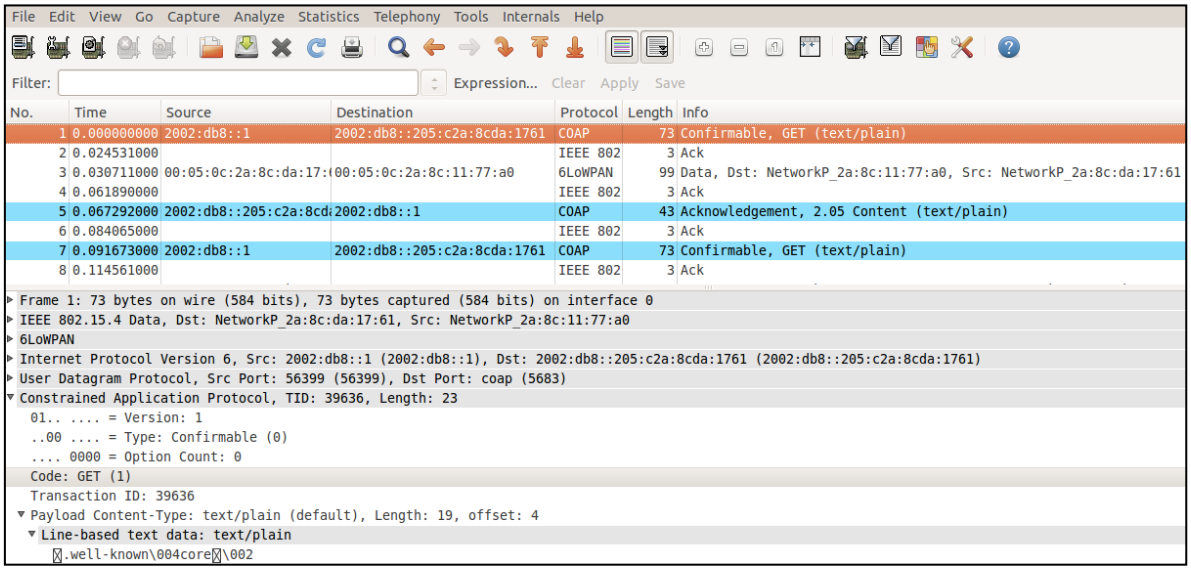


Fig. 81 Format del paquet GET

GET: "Hello World!"

Fig. 82 Petició GET: Hello World

Format del paquet CoAP "hello world"

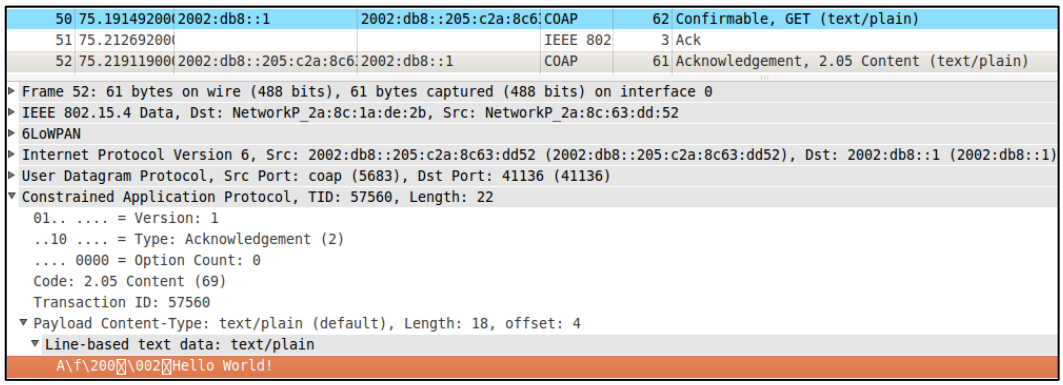


Fig. 83 Format GET: Hello World

GET: Button

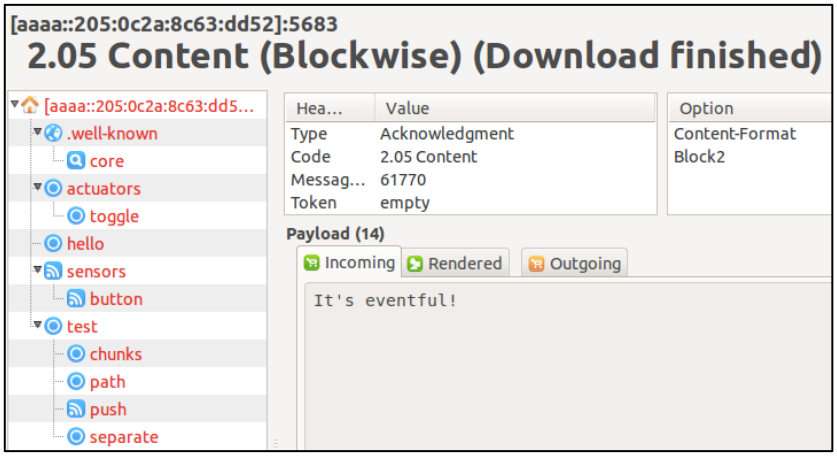


Fig. 84 Petició GET: Button

GET Button

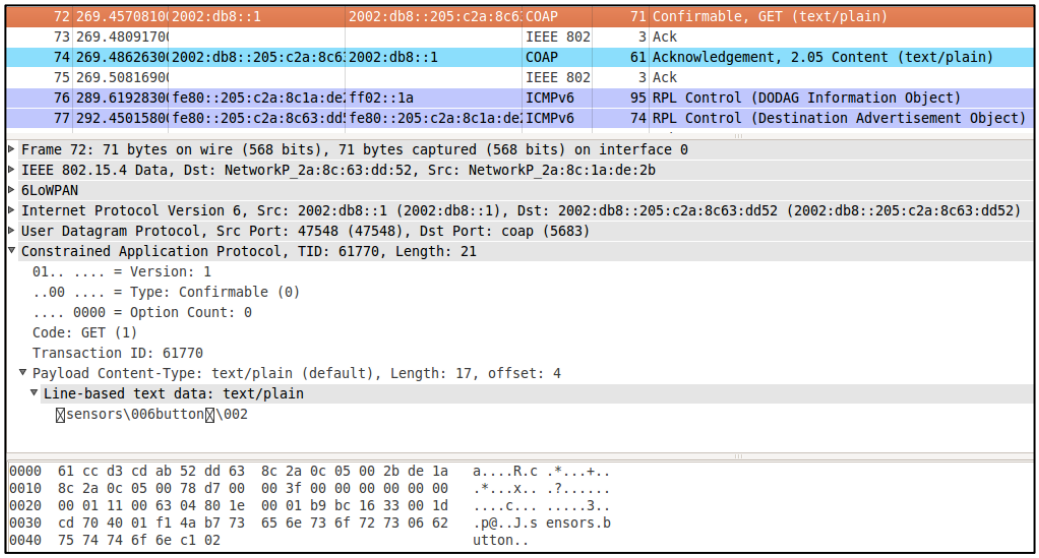


Fig. 85 Format GET Button

ACK Button

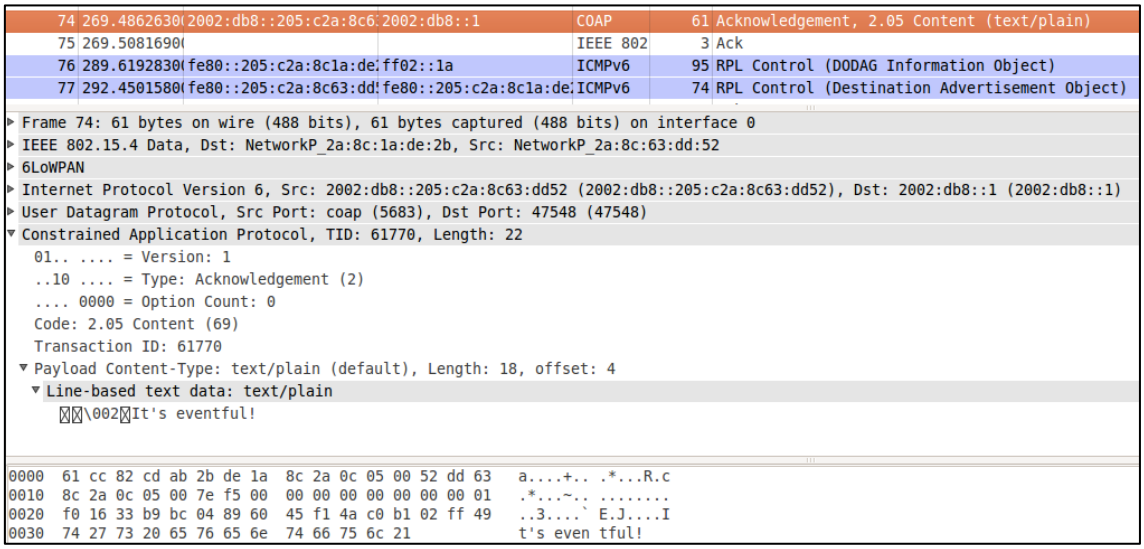


Fig. 86 Resposta GET: Button

6.2.2. Prova Opció OBSERVE

La opció Observe de CoAP, s'ha vist en l'apartat de teoria: el client envia una petició GET al servidor amb la opció 'Observe' activa.
El servidor enviarà les notificacions corresponents als estats d'aquest recurs, per notificar així en cas de canvis.

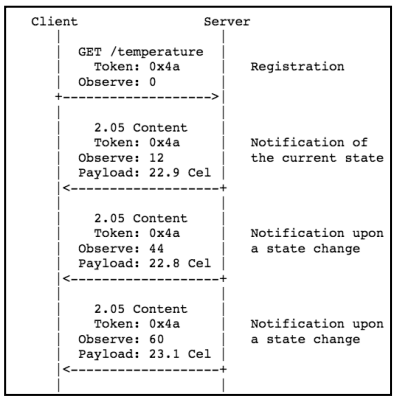


Fig. 87 OBSERVE CoAP

El Recurs botó (*button*) permet Observe: Sensors -> button

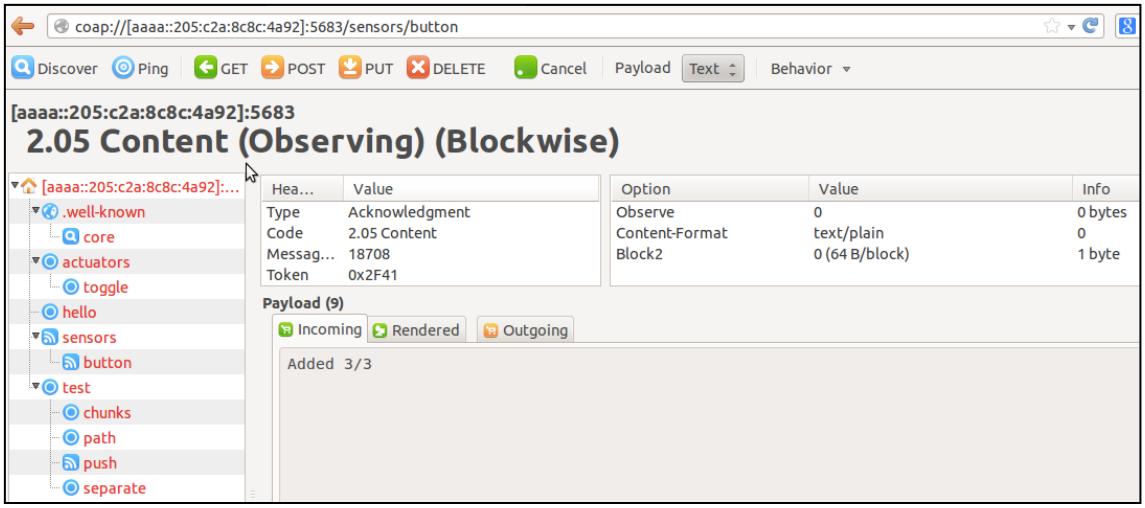


Fig. 88 Observe del Recurs button

Format del paquet Observe: GET amb dues opcions, observe i block2

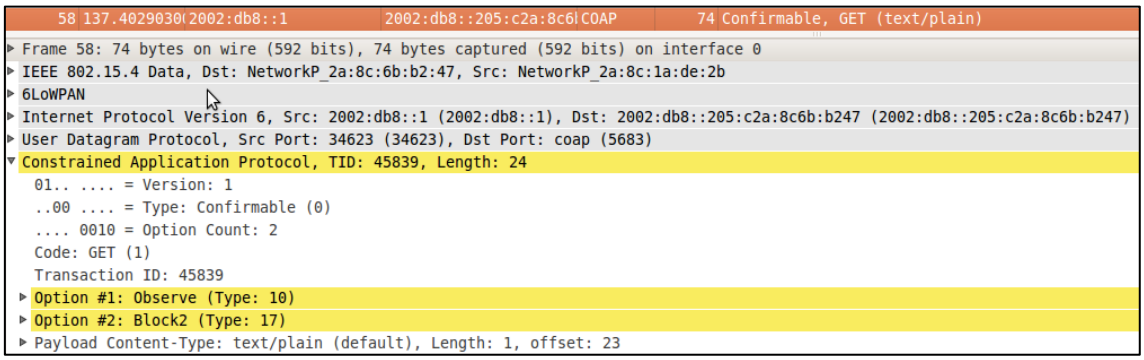


Fig. 89 Paquet GET amb Observe Option

Observe recurs Push: el value Tick es va incrementant

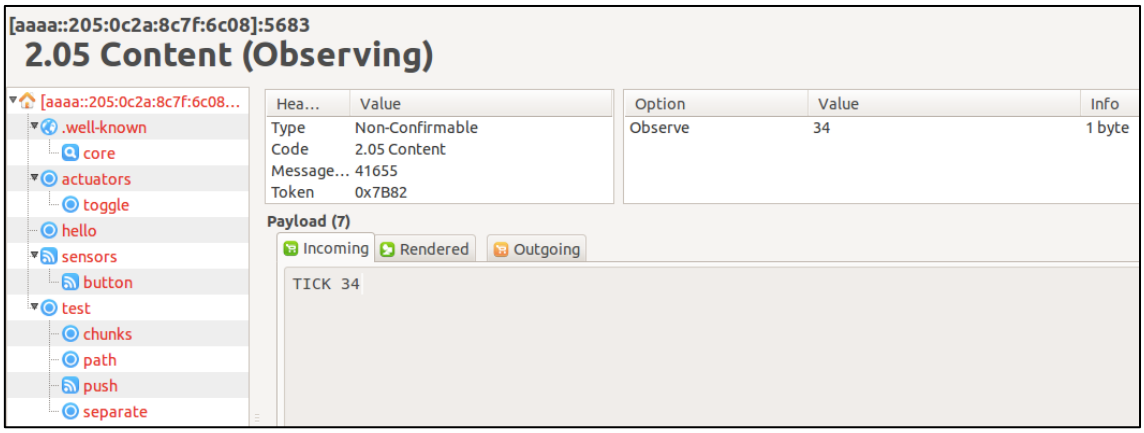


Fig. 90 Observe: Push

Respostes del servidor: non-confirmables

No.	Time	Source	Destination	Protocol	Length	Info
61	178.8284350(	2002:db8::205:c2a:8c7f:6c08	2002:db8::1	COAP	59	Acknowledgement, 2.05 Content
62	178.8494930(			IEEE 802	3	Ack
63	179.1152160(	2002:db8::205:c2a:8c7f:6c08	2002:db8::1	COAP	55	Non-Confirmable, 2.05 Content
64	179.1358580(			IEEE 802	3	Ack
65	184.1047150(	2002:db8::205:c2a:8c7f:6c08	2002:db8::1	COAP	55	Non-Confirmable, 2.05 Content
66	184.1243260(			IEEE 802	3	Ack
67	189.0938230(	2002:db8::205:c2a:8c7f:6c08	2002:db8::1	COAP	55	Non-Confirmable, 2.05 Content
68	189.1128750(			IEEE 802	3	Ack
69	194.0816520(	2002:db8::205:c2a:8c7f:6c08	2002:db8::1	COAP	55	Non-Confirmable, 2.05 Content
70	194.1012310(			IEEE 802	3	Ack
71	199.0713840(	2002:db8::205:c2a:8c7f:6c08	2002:db8::1	COAP	55	Non-Confirmable, 2.05 Content
72	199.0904720(			IEEE 802	3	Ack
73	204.0593100(	2002:db8::205:c2a:8c7f:6c08	2002:db8::1	COAP	55	Non-Confirmable, 2.05 Content
74	204.0786660(			IEEE 802	3	Ack
75	209.0484340(	2002:db8::205:c2a:8c7f:6c08	2002:db8::1	COAP	55	Non-Confirmable, 2.05 Content
76	209.0679320(			IEEE 802	3	Ack

Fig. 91 Paquets Non-Confirmables

6.3. Escenaris amb múltiples Nodes

Escenari REST amb tres nodes: 1 Border Router + 2 Servidors CoAP

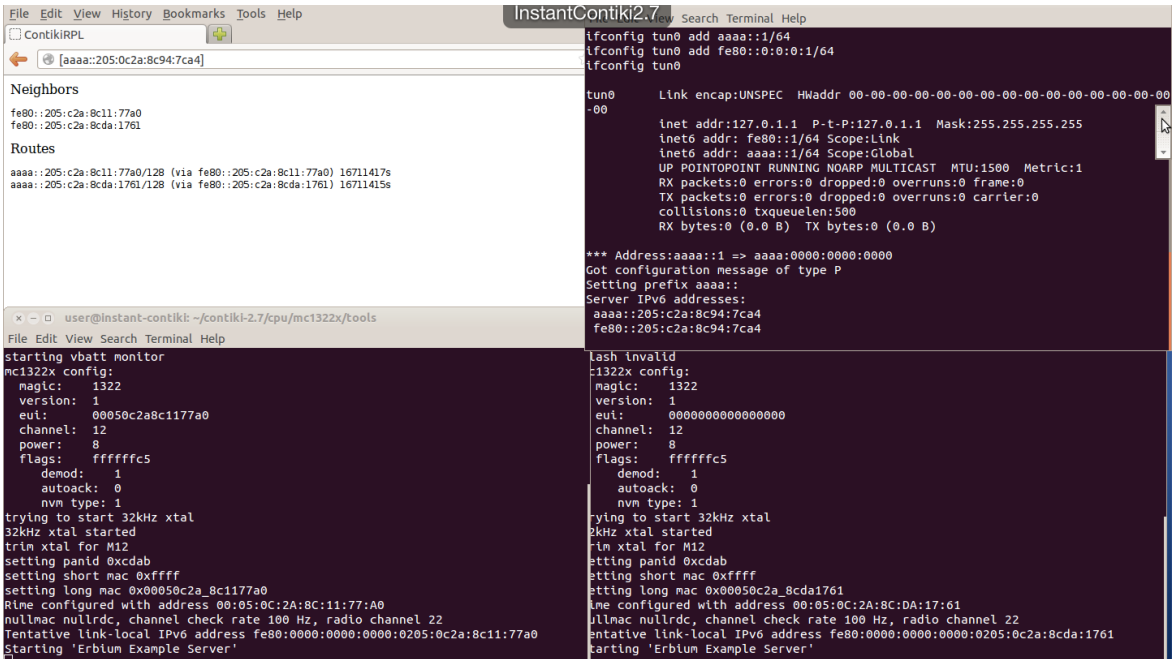


Fig. 92 Escenari WSN amb 3 nodes

Escenari amb quatre nodes executant:

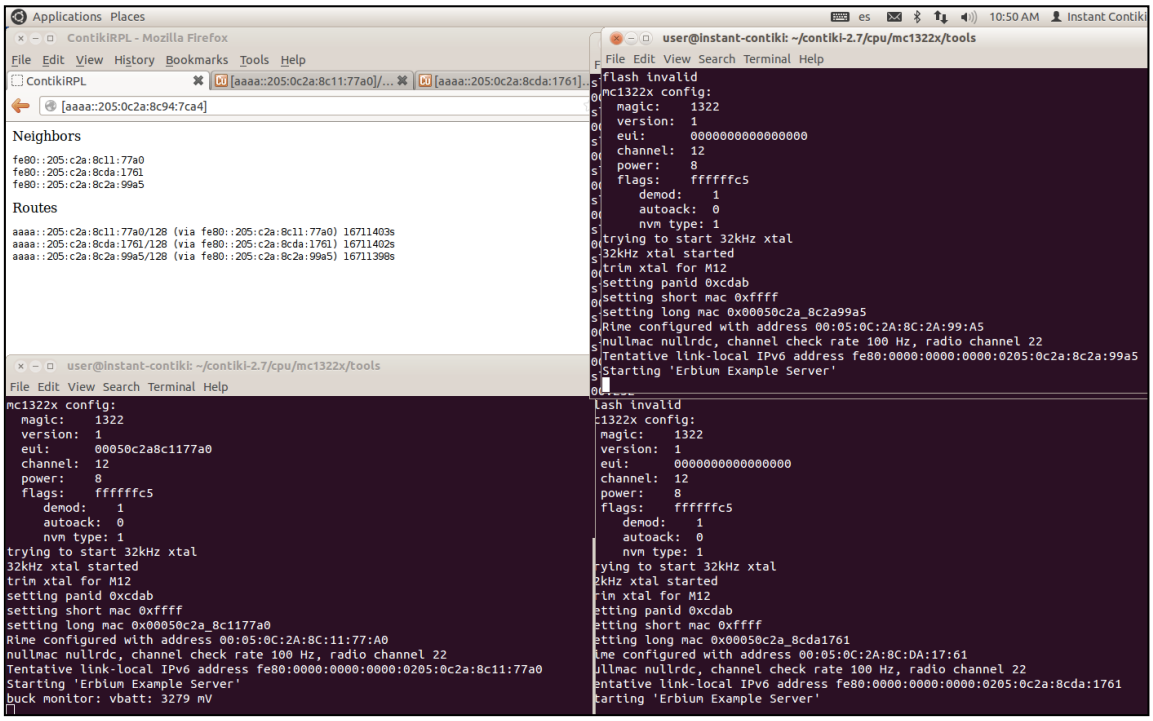


Fig. 93 Escenari WSN amb 4 nodes

6.4. Configuració de Capa MAC: Modes RDC

6.4.1. NullRDC_driver

Formació DODAG

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::205:c2a:8c6b:b2	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
2	13.41277900	fe80::205:c2a:8c6b:b2	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
3	20.44643100	fe80::205:c2a:8c6b:b2	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
4	33.78588400	fe80::205:c2a:8c6b:b2	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
5	36.55837400	fe80::205:c2a:8c63:dd	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
6	37.47480100	fe80::205:c2a:8c63:dd	fe80::205:c2a:8c6b:b2	ICMPv6	74	RPL Control (Destination Advertisement Object)
7	37.49920900			IEEE 802	3	Ack
8	45.52805200	fe80::205:c2a:8c63:dd	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)

Fig. 94 Formació DODAG NullRDC

Ping router (20,4 ms)

```
--- aaaa::0205:0c2a:8c1a:de2b ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 19.378/20.362/20.995/0.583 ms
```

Fig. 95 Ping RTT router

Ping a la mota a un salt

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	00:05:0c:2a:8c:6b:b2:	00:05:0c:2a:8c:63:dd:	6LoWPAN	102	Data, Dst: NetworkP_2a:8c:63:dd:52, Src: NetworkP_2a:8c:6b:b2:47
2	0.035777000			IEEE 802	3	Ack
3	0.037333000	2002:db8::1	2002:db8::205:c2a:8c6	ICMPv6	34	Echo (ping) request id=0x0ad0, seq=1
4	0.050716000			IEEE 802	3	Ack
5	0.056816000	00:05:0c:2a:8c:63:dd:	00:05:0c:2a:8c:6b:b2:	6LoWPAN	101	Data, Dst: NetworkP_2a:8c:6b:b2:47, Src: NetworkP_2a:8c:63:dd:52
6	0.092937000			IEEE 802	3	Ack
7	0.095981000	2002:db8::205:c2a:8c6	2002:db8::1	ICMPv6	34	Echo (ping) reply id=0x0ad0, seq=1
8	0.107999000			IEEE 802	3	Ack
9	1.000435000	00:05:0c:2a:8c:6b:b2:	00:05:0c:2a:8c:63:dd:	6LoWPAN	102	Data, Dst: NetworkP_2a:8c:63:dd:52, Src: NetworkP_2a:8c:6b:b2:47
10	1.033898000			IEEE 802	3	Ack
11	1.038468000	2002:db8::1	2002:db8::205:c2a:8c6	ICMPv6	34	Echo (ping) request id=0x0ad0, seq=2
12	1.052541000			IEEE 802	3	Ack

Fig. 96 Ping a un salt Wireshark

RTT = 37,7 ms

```
user@instant-contiki:~$ ping6 aaaa::205:0c2a:8c63:dd52
PING aaaa::205:0c2a:8c63:dd52(aaaa::205:c2a:8c63:dd52) 56 data bytes
64 bytes from aaaa::205:c2a:8c63:dd52: icmp_seq=1 ttl=63 time=38.3 ms
64 bytes from aaaa::205:c2a:8c63:dd52: icmp_seq=2 ttl=63 time=36.3 ms
64 bytes from aaaa::205:c2a:8c63:dd52: icmp_seq=3 ttl=63 time=37.0 ms
64 bytes from aaaa::205:c2a:8c63:dd52: icmp_seq=4 ttl=63 time=38.1 ms
64 bytes from aaaa::205:c2a:8c63:dd52: icmp_seq=5 ttl=63 time=38.6 ms
^C
--- aaaa::205:0c2a:8c63:dd52 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 36.332/37.727/38.654/0.877 ms
```

Fig. 97 RTT NullRDC a un salt

6.4.2. ContikiMAC_driver

contikimac_driver = contikimac-RDC

Amb el mode ContikiMAC activat, els missatges de RPL que s'intercanvien les dues motes, segueixen aquest esquema:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::205:c2a:8c63:dd52	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
2	0.901090000	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
3	0.934423000	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
4	0.968980000	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
5	8.444475000	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
6	8.475230000	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
7	8.506809000	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
8	19.36956000	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
9	19.40045300	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
10	19.43245700	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
11	48.01598900	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
12	48.04615300	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
13	48.07757700	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
14	59.86572500	fe80::205:c2a:8c63:dd52	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
15	62.54351500	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
16	62.57413200	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
17	62.60634900	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
18	71.98220900	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
19	72.01508100	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
20	72.04402100	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
21	88.40470000	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
22	88.43960700	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
23	88.46600100	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
24	108.6990640	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
25	108.7326250	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
26	108.7610970	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47
27	119.7305400	fe80::205:c2a:8c63:dd52	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
28	122.6674480	00:05:0c:2a:8c:6b:b2:47	Broadcast	IEEE 802	97	Data, Dst: Broadcast, Src: NetworkP_2a:8c:6b:b2:47

Fig. 98 Missatges RPL amb ContikiMAC-RDC activat

Sembla que aquest mode no està del tot suportat pels nostres dispositius, els Econotags.

No puc fer ping a la mota, no es veuen.

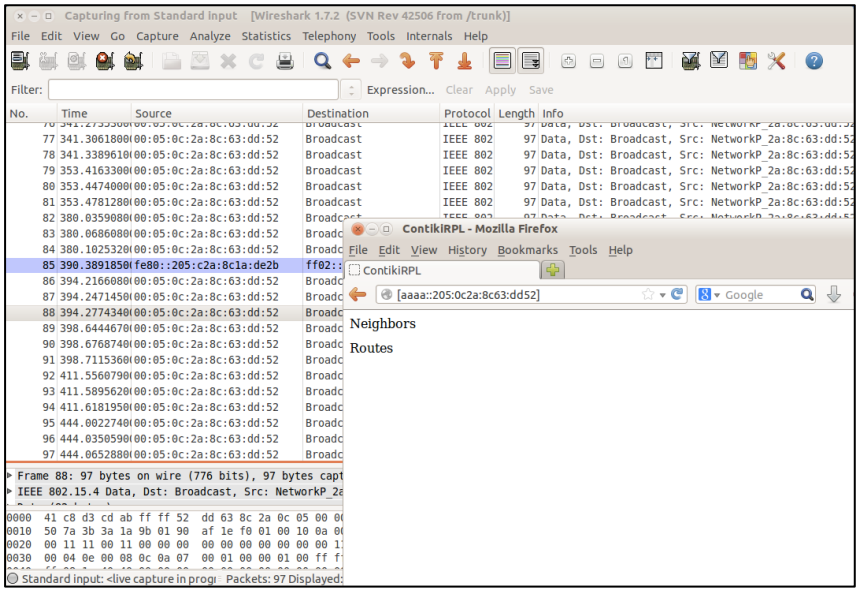


Fig. 99 ContikiMAC-RDC, Border Router no anuncia la mota

6.4.3. SicslowMAC

Router (20,7 ms)

```
user@instant-contiki:~$ ping6 aaaa::205:0c2a:8c14:d607
PING aaaa::205:0c2a:8c14:d607(aaaa::205:c2a:8c14:d607) 56 data bytes
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=1 ttl=64 time=20.3 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=2 ttl=64 time=20.6 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=3 ttl=64 time=19.6 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=4 ttl=64 time=20.4 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=5 ttl=64 time=21.3 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=6 ttl=64 time=21.6 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=7 ttl=64 time=19.4 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=8 ttl=64 time=21.2 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=9 ttl=64 time=21.0 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=10 ttl=64 time=20.9 ms
^C
--- aaaa::205:0c2a:8c14:d607 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9013ms
rtt min/avg/max/mdev = 19.476/20.677/21.643/0.697 ms
```

Fig. 100 Ping Router Sicslowmac

Mota (34,1 ms)

```
user@instant-contiki:~$ ping6 aaaa::205:c2a:8c1a:de2b
PING aaaa::205:c2a:8c1a:de2b(aaaa::205:c2a:8c1a:de2b) 56 data bytes
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=1 ttl=63 time=35.1 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=2 ttl=63 time=34.1 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=3 ttl=63 time=33.5 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=4 ttl=63 time=33.5 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=5 ttl=63 time=34.2 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=6 ttl=63 time=34.2 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=7 ttl=63 time=33.6 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=8 ttl=63 time=34.4 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=9 ttl=63 time=34.0 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=10 ttl=63 time=34.0 ms
^C
--- aaaa::205:c2a:8c1a:de2b ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 33.553/34.107/35.114/0.454 ms
```

Fig. 101 Ping mota sicslowmac

6.4.4. LPP_driver

Ping Router: 20,4 ms

```
PING aaaa::205:0c2a:8c14:d607(aaaa::205:c2a:8c14:d607) 56 data bytes
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=1 ttl=64 time=20.6 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=2 ttl=64 time=20.4 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=3 ttl=64 time=20.7 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=4 ttl=64 time=19.7 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=5 ttl=64 time=21.0 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=6 ttl=64 time=20.0 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=7 ttl=64 time=20.3 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=8 ttl=64 time=20.2 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=9 ttl=64 time=19.8 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=10 ttl=64 time=21.1 ms
^C
--- aaaa::205:0c2a:8c14:d607 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9015ms
rtt min/avg/max/mdev = 19.703/20.413/21.112/0.466 ms
```

Fig. 102 Ping Router lpp

Ping mota: 37,5 ms

```
PING aaaa::205:0c2a:8c1a:de2b(aaaa::205:c2a:8c1a:de2b) 56 data bytes
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=1 ttl=63 time=37.3 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=2 ttl=63 time=38.2 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=3 ttl=63 time=36.9 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=4 ttl=63 time=37.8 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=5 ttl=63 time=37.8 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=6 ttl=63 time=37.2 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=7 ttl=63 time=37.3 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=8 ttl=63 time=37.5 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=9 ttl=63 time=36.9 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=10 ttl=63 time=37.4 ms
^C
--- aaaa::205:0c2a:8c1a:de2b ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 36.946/37.494/38.279/0.458 ms
```

Fig. 103 Ping mota lpp

6.4.5. CXMAC_driver

Router (20,07 ms)

```
user@instant-contiki:~$ ping6 aaaa::205:0c2a:8c14:d607
PING aaaa::205:0c2a:8c14:d607(aaaa::205:c2a:8c14:d607) 56 data bytes
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=1 ttl=64 time=20.7 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=2 ttl=64 time=20.1 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=3 ttl=64 time=19.5 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=4 ttl=64 time=20.4 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=5 ttl=64 time=19.6 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=6 ttl=64 time=19.6 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=7 ttl=64 time=20.6 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=8 ttl=64 time=20.6 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=9 ttl=64 time=19.7 ms
64 bytes from aaaa::205:c2a:8c14:d607: icmp_seq=10 ttl=64 time=19.5 ms
^C
--- aaaa::205:0c2a:8c14:d607 ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9016ms
rtt min/avg/max/mdev = 19.519/20.073/20.770/0.511 ms
```

Fig. 104 Ping Router CXMAC

Apareix el node veí, però no s'anuncia la ruta "aaaa::205:c2a:8c1a:de2b"

Fig. 105 ContikiRPL CXMAC

Prova ping mota:

```
slip-bridge: Destination off-link but no route src=aaaa::1 dst=aaaa::205:c2a:8c1
a:de2b
slip-bridge: Destination off-link but no route src=aaaa::1 dst=aaaa::205:c2a:8c1
a:de2b
slip-bridge: Destination off-link but no route src=aaaa::1 dst=aaaa::205:c2a:8c1
a:de2b
slip-bridge: Destination off-link but no route src=aaaa::1 dst=aaaa::205:c2a:8c1
a:de2b
```

Fig. 106 Ping mota CXMAC

6.5. Configuració de capa MAC: CSMA vs NullMAC

CSMA: average RTT = 39,064 ms

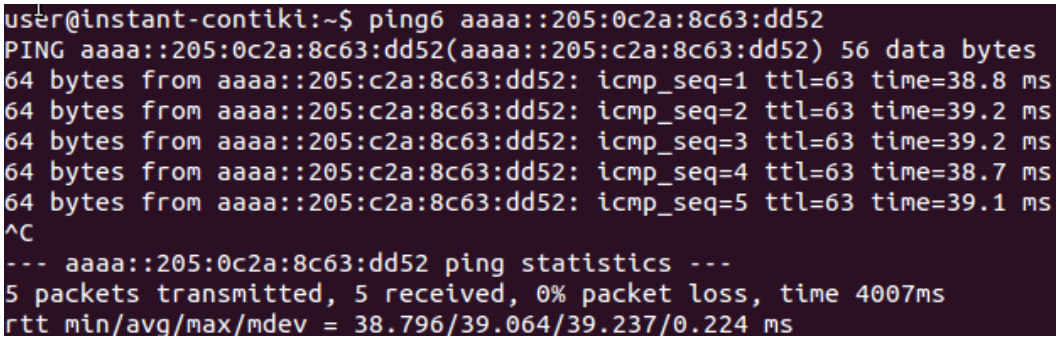


Fig. 107 Ping amb CSMA

NullMAC: average RTT = 37,727 ms

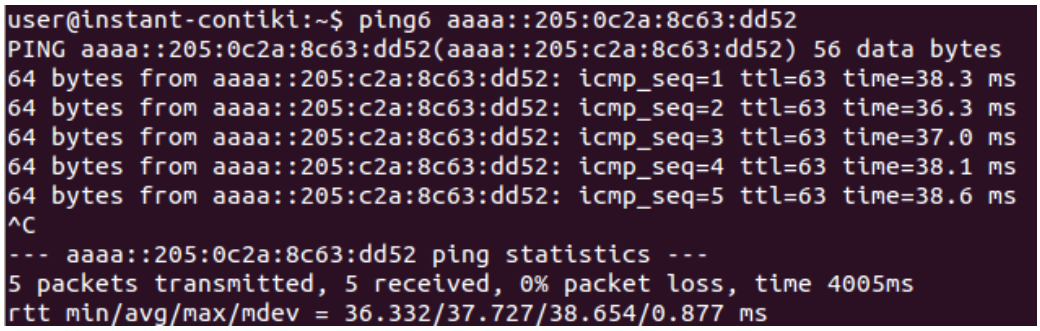


Fig. 108 Ping NullMAC (sense escoltar)

6.6. Configuració de capa MAC: ACK's vs Sense ACK's

Formació DAG amb Acks, deprés dels missatges DAO hi ha Acks

5	422.289154000	fe80::205:c2a:8c2a:99	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
6	425.392136000	fe80::205:c2a:8c34:b1	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
7	427.869478000	fe80::205:c2a:8c2a:99	fe80::205:c2a:8c34:b1	ICMPv6	74	RPL Control (Destination Advertisement Object)
8	427.893900000			IEEE 802	3	Ack
9	428.427966000	fe80::205:c2a:8c2a:99	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
10	430.640499000	fe80::205:c2a:8c34:b1	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
11	434.484839000	fe80::205:c2a:8c2a:99	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
12	434.704366000	fe80::205:c2a:8c2a:99	fe80::205:c2a:8c34:b1	ICMPv6	74	RPL Control (Destination Advertisement Object)
13	434.728077000			IEEE 802	3	Ack

Fig. 109 Formació DAG amb ACKs

Formació DAG sense Acks

2	30.51129600	fe80::205:c2a:8c2a:99	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
3	33.30488800	fe80::205:c2a:8c14:d6	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
4	35.91106000	fe80::205:c2a:8c2a:99	fe80::205:c2a:8c14:d6	ICMPv6	74	RPL Control (Destination Advertisement Object)
5	36.32139000	fe80::205:c2a:8c2a:99	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
6	40.93711000	fe80::205:c2a:8c14:d6	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
7	43.33422100	fe80::205:c2a:8c2a:99	fe80::205:c2a:8c14:d6	ICMPv6	74	RPL Control (Destination Advertisement Object)
8	43.74330000	fe80::205:c2a:8c2a:99	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)

Fig. 110 Formació DAG sense ACKs

Ping mota amb Acks

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	00:05:0c:2a:8c:34:b1:c	00:05:0c:2a:8c:2a:99:a	6LoWPAN	102	Data, Dst: NetworkP_2a:8c:2a:99:a5, Src: NetworkP_2a:8c:34:b1:c7
2	0.032414000			IEEE 802	3	Ack
3	0.038582000	2002:db8::1	2002:db8::205:c2a:8c2	ICMPv6	34	Echo (ping) request id=0x0db3, seq=1
4	0.052726000			IEEE 802	3	Ack
5	0.056208000	00:05:0c:2a:8c:2a:99:a	00:05:0c:2a:8c:34:b1:c	6LoWPAN	101	Data, Dst: NetworkP_2a:8c:34:b1:c7, Src: NetworkP_2a:8c:2a:99:a5
6	0.088594000			IEEE 802	3	Ack
7	0.094556000	2002:db8::205:c2a:8c2	2002:db8::1	ICMPv6	34	Echo (ping) reply id=0x0db3, seq=1
8	0.108430000			IEEE 802	3	Ack
9	1.002870000	00:05:0c:2a:8c:34:b1:c	00:05:0c:2a:8c:2a:99:a	6LoWPAN	102	Data, Dst: NetworkP_2a:8c:2a:99:a5, Src: NetworkP_2a:8c:34:b1:c7
10	1.037541000			IEEE 802	3	Ack
11	1.040078000	2002:db8::1	2002:db8::205:c2a:8c2	ICMPv6	34	Echo (ping) request id=0x0db3, seq=2
12	1.054046000			IEEE 802	3	Ack
13	1.061645000	00:05:0c:2a:8c:2a:99:a	00:05:0c:2a:8c:34:b1:c	6LoWPAN	101	Data, Dst: NetworkP_2a:8c:34:b1:c7, Src: NetworkP_2a:8c:2a:99:a5
14	1.091538000			IEEE 802	3	Ack
15	1.100410000	2002:db8::205:c2a:8c2	2002:db8::1	ICMPv6	34	Echo (ping) reply id=0x0db3, seq=2
16	1.112101000			IEEE 802	3	Ack
17	2.005283000	00:05:0c:2a:8c:34:b1:c	00:05:0c:2a:8c:2a:99:a	6LoWPAN	102	Data, Dst: NetworkP_2a:8c:2a:99:a5, Src: NetworkP_2a:8c:34:b1:c7
18	2.037866000			IEEE 802	3	Ack

Fig. 111 Ping a la mota amb ACKs

Ping mota sense Acks (hi ha menys retard que a dalt)

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	00:05:0c:2a:8c:14:d6:0	00:05:0c:2a:8c:2a:99:a	6LoWPAN	102	Data, Dst: NetworkP_2a:8c:2a:99:a5, Src: NetworkP_2a:8c:14:d6:07
2	0.031325000	2002:db8::1	2002:db8::205:c2a:8c2	ICMPv6	34	Echo (ping) request id=0x0ba5, seq=1
3	0.046087000	00:05:0c:2a:8c:2a:99:a	00:05:0c:2a:8c:14:d6:0	6LoWPAN	101	Data, Dst: NetworkP_2a:8c:14:d6:07, Src: NetworkP_2a:8c:2a:99:a5
4	0.081987000	2002:db8::205:c2a:8c2	2002:db8::1	ICMPv6	34	Echo (ping) reply id=0x0ba5, seq=1
5	1.001771000	00:05:0c:2a:8c:14:d6:0	00:05:0c:2a:8c:2a:99:a	6LoWPAN	102	Data, Dst: NetworkP_2a:8c:2a:99:a5, Src: NetworkP_2a:8c:14:d6:07
6	1.035408000	2002:db8::1	2002:db8::205:c2a:8c2	ICMPv6	34	Echo (ping) request id=0x0ba5, seq=2
7	1.048785000	00:05:0c:2a:8c:2a:99:a	00:05:0c:2a:8c:14:d6:0	6LoWPAN	101	Data, Dst: NetworkP_2a:8c:14:d6:07, Src: NetworkP_2a:8c:2a:99:a5
8	1.080508000	2002:db8::205:c2a:8c2	2002:db8::1	ICMPv6	34	Echo (ping) reply id=0x0ba5, seq=2
9	2.002172000	00:05:0c:2a:8c:14:d6:0	00:05:0c:2a:8c:2a:99:a	6LoWPAN	102	Data, Dst: NetworkP_2a:8c:2a:99:a5, Src: NetworkP_2a:8c:14:d6:07
10	2.035969000	2002:db8::1	2002:db8::205:c2a:8c2	ICMPv6	34	Echo (ping) request id=0x0ba5, seq=3
11	2.048790000	00:05:0c:2a:8c:2a:99:a	00:05:0c:2a:8c:14:d6:0	6LoWPAN	101	Data, Dst: NetworkP_2a:8c:14:d6:07, Src: NetworkP_2a:8c:2a:99:a5

Fig. 112 Ping a la mota sense ACKs

6.7. Configuració de capa MAC: RTT ACK's vs Sense ACK's

- Border Router (salt 0) : 20 ms
- Ping mota (sense acks): 34 ms

```
PING aaaa::205:0c2a:8c1a:de2b(aaaa::205:c2a:8c1a:de2b) 56 data bytes
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=1 ttl=63 time=33.3 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=2 ttl=63 time=33.7 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=3 ttl=63 time=34.3 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=4 ttl=63 time=34.0 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=5 ttl=63 time=33.5 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=6 ttl=63 time=34.2 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=7 ttl=63 time=34.6 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=8 ttl=63 time=33.4 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=9 ttl=63 time=34.2 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=10 ttl=63 time=34.1 ms
^C
--- aaaa::205:0c2a:8c1a:de2b ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9016ms
rtt min/avg/max/mdev = 33.377/33.993/34.619/0.435 ms
```

Fig. 113 RTT a un salt sense acks

- Ping mota a dos salts (sense acks): 49,2 ms

```
user@instant-contiki:~$ ping6 aaaa::205:c2a:8c2a:99a5
PING aaaa::205:c2a:8c2a:99a5(aaaa::205:c2a:8c2a:99a5) 56 data bytes
64 bytes from aaaa::205:c2a:8c2a:99a5: icmp_seq=5 ttl=62 time=49.3 ms
64 bytes from aaaa::205:c2a:8c2a:99a5: icmp_seq=6 ttl=62 time=49.0 ms
64 bytes from aaaa::205:c2a:8c2a:99a5: icmp_seq=7 ttl=62 time=48.6 ms
64 bytes from aaaa::205:c2a:8c2a:99a5: icmp_seq=13 ttl=62 time=49.6 ms
64 bytes from aaaa::205:c2a:8c2a:99a5: icmp_seq=14 ttl=62 time=49.3 ms
^C
--- aaaa::205:c2a:8c2a:99a5 ping statistics ---
15 packets transmitted, 5 received, 66% packet loss, time 14078ms
rtt min/avg/max/mdev = 48.685/49.224/49.646/0.388 ms
```

Fig. 114 RTT a dos sants sense acks

- Ping mota (amb acks): 37,9 ms

```
PING aaaa::205:0c2a:8c1a:de2b(aaaa::205:c2a:8c1a:de2b) 56 data bytes
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=1 ttl=63 time=38.2 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=2 ttl=63 time=37.1 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=3 ttl=63 time=38.2 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=4 ttl=63 time=38.3 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=5 ttl=63 time=37.4 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=6 ttl=63 time=36.8 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=7 ttl=63 time=37.9 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=8 ttl=63 time=38.2 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=9 ttl=63 time=38.0 ms
64 bytes from aaaa::205:c2a:8c1a:de2b: icmp_seq=10 ttl=63 time=38.3 ms
^C
--- aaaa::205:0c2a:8c1a:de2b ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9014ms
rtt min/avg/max/mdev = 36.890/37.901/38.385/0.524 ms
```

Fig. 115 RTT a un salt amb acks

6.8. Configuració de RPL

6.8.1. Proves Imin

Amb un node:

Imin = 2^8 = 0,256 seg

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
2	6.676902000	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
3	13.61195200(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
4	27.07135200(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
5	63.03966900(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
6	107.3203480(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
7	229.0151290(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
8	419.6050000(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)

Fig. 116 RPL Imin=8

Imin = 2^{11} = 2,048 seg

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
2	7.423843000	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
3	18.04979000(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
4	50.57587300(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
5	112.8962290(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
6	211.6031490(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
7	455.9620660(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)

Fig. 117 RPL Imin = 11

Imin = 2^{12} = 4,096 seg

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
2	11.74656100(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
3	19.20890500(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
4	31.08178200(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
5	63.08024700(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
6	132.8327730(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
7	239.2223490(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
8	441.6644540(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)

Fig. 118 RPL Imin = 12

Proves Imin amb dos nodes

Imin = 11 : 2,048 seg

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
2	7.843965000	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
3	15.17802300(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
4	28.23801300(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
5	55.22689800(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
6	93.80354900(	fe80::205:c2a:8c7f:6c08	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
7	96.63429600(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
8	99.06389600(	fe80::205:c2a:8c7f:6c08	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
9	100.5797240(	fe80::205:c2a:8c7f:6c08	fe80::205:c2a:8c1a:de2	ICMPv6	74	RPL Control (Destination Advertisement Object)
10	100.6040160(			IEEE 802	3	Ack
11	103.6480660(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
12	108.2037340(	fe80::205:c2a:8c7f:6c08	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
13	117.7959030(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
14	122.2114640(	fe80::205:c2a:8c7f:6c08	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
15	123.1285710(	fe80::205:c2a:8c7f:6c08	fe80::205:c2a:8c1a:de2	ICMPv6	74	RPL Control (Destination Advertisement Object)
16	123.1539710(			IEEE 802	3	Ack
17	152.1890020(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
18	157.0530910(	fe80::205:c2a:8c7f:6c08	fe80::205:c2a:8c1a:de2	ICMPv6	74	RPL Control (Destination Advertisement Object)
19	157.0782430(			IEEE 802	3	Ack
20	157.2927710(	fe80::205:c2a:8c7f:6c08	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
21	196.1660680(	fe80::205:c2a:8c7f:6c08	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
22	206.3368840(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
23	209.6556630(	fe80::205:c2a:8c7f:6c08	fe80::205:c2a:8c1a:de2	ICMPv6	74	RPL Control (Destination Advertisement Object)
24	209.6797360(			IEEE 802	3	Ack
25	319.3007710(	fe80::205:c2a:8c7f:6c08	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
26	321.2276780(	fe80::205:c2a:8c1a:de2b	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
27	323.4097470(	fe80::205:c2a:8c7f:6c08	fe80::205:c2a:8c1a:de2	ICMPv6	74	RPL Control (Destination Advertisement Object)
28	323.4347680(			IEEE 802	3	Ack

Fig. 119 Imin = 11 Dos nodes

Imin = 12 : 2^{12} = 4,096 seg

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::205:c2a:8c1a:de2	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
2	2.218617000	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
3	9.192284000	fe80::205:c2a:8c1a:de2	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
4	11.66935900(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
5	13.10638400(	fe80::205:c2a:8c7f:6c(	fe80::205:c2a:8c1a:de2	ICMPv6	74	RPL Control (Destination Advertisement Object)
6	13.13033600(			IEEE 802	3	Ack
7	14.95914700(	fe80::205:c2a:8c1a:de2	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
8	17.13715400(	fe80::205:c2a:8c7f:6c(	fe80::205:c2a:8c1a:de2	ICMPv6	74	RPL Control (Destination Advertisement Object)
9	17.16132200(			IEEE 802	3	Ack
10	18.35478100(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
11	30.09488200(	fe80::205:c2a:8c1a:de2	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
12	35.05616400(	fe80::205:c2a:8c7f:6c(	fe80::205:c2a:8c1a:de2	ICMPv6	74	RPL Control (Destination Advertisement Object)
13	35.08164100(			IEEE 802	3	Ack
14	37.15353700(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
15	53.64125600(	fe80::205:c2a:8c1a:de2	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
16	58.42335100(	fe80::205:c2a:8c7f:6c(	fe80::205:c2a:8c1a:de2	ICMPv6	74	RPL Control (Destination Advertisement Object)
17	58.44819600(			IEEE 802	3	Ack
18	69.33979300(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
19	107.0311830(	fe80::205:c2a:8c1a:de2	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
20	111.7445990(	fe80::205:c2a:8c7f:6c(	fe80::205:c2a:8c1a:de2	ICMPv6	74	RPL Control (Destination Advertisement Object)
21	111.7687700(			IEEE 802	3	Ack
22	125.4742210(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
23	210.1189400(	fe80::205:c2a:8c1a:de2	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
24	215.6218900(	fe80::205:c2a:8c7f:6c(	fe80::205:c2a:8c1a:de2	ICMPv6	74	RPL Control (Destination Advertisement Object)
25	215.6459260(			IEEE 802	3	Ack
26	244.5877600(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)

Fig. 120 Imin = 12 Dos nodes

6.8.2. Proves Constant k

Missatges RPL amb k=1:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
2	7.904358000	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
3	16.18560200(	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
4	25.95301100(	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
5	28.24912800(	fe80::205:c2a:8c1a:de:	fe80::205:c2a:8c14:d6(	ICMPv6	74	RPL Control (Destination Advertisement Object)
6	28.27411500(			IEEE 802	3	Ack
7	28.37999300(	fe80::205:c2a:8c1a:de:	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
8	34.28606600(	fe80::205:c2a:8c1a:de:	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
9	46.83790800(	fe80::205:c2a:8c1a:de:	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
10	52.86223400(	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
11	57.16373500(	fe80::205:c2a:8c1a:de:	fe80::205:c2a:8c14:d6(	ICMPv6	74	RPL Control (Destination Advertisement Object)
12	57.18888500(			IEEE 802	3	Ack
13	87.21722000(	fe80::205:c2a:8c1a:de:	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
14	108.2778340(	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
15	111.4415820(	fe80::205:c2a:8c1a:de:	fe80::205:c2a:8c14:d6(	ICMPv6	74	RPL Control (Destination Advertisement Object)
16	111.4656820(			IEEE 802	3	Ack
17	133.5119060(	fe80::205:c2a:8c1a:de:	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
18	207.9146750(	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
19	212.3934680(	fe80::205:c2a:8c1a:de:	fe80::205:c2a:8c14:d6(	ICMPv6	74	RPL Control (Destination Advertisement Object)
20	212.4180970(			IEEE 802	3	Ack

Fig. 121 RPL k=1

Missatges RPL amb k=2

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
2	5.857098000	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
3	18.40951800(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
4	25.88699400(	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
5	29.67425000(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
6	31.95540400(	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
7	32.28307200(	fe80::205:c2a:8c14:d6(	fe80::205:c2a:8c7f:6c(	ICMPv6	74	RPL Control (Destination Advertisement Object)
8	32.30866500(			IEEE 802	3	Ack
9	37.30694100(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
10	40.56600500(	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
11	42.76084600(	fe80::205:c2a:8c14:d6(	fe80::205:c2a:8c7f:6c(	ICMPv6	74	RPL Control (Destination Advertisement Object)
12	42.78519400(			IEEE 802	3	Ack
13	48.88052200(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
14	51.76108100(	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
15	51.79352400(	fe80::205:c2a:8c14:d6(	fe80::205:c2a:8c7f:6c(	ICMPv6	74	RPL Control (Destination Advertisement Object)
16	51.81591500(			IEEE 802	3	Ack
17	71.80918200(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
18	75.41698100(	fe80::205:c2a:8c14:d6(	fe80::205:c2a:8c7f:6c(	ICMPv6	74	RPL Control (Destination Advertisement Object)
19	75.44184900(			IEEE 802	3	Ack

Fig. 122 RPL k=2

Missatges RPL amb k=10

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
2	6.834714000	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
3	9.159712000	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	25	RPL Control (DODAG Information Solicitation)
4	11.76320700(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
5	14.28047600(	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
6	16.47432500(	fe80::205:c2a:8c14:d6(	fe80::205:c2a:8c7f:6c(	ICMPv6	74	RPL Control (Destination Advertisement Object)
7	16.49887900(			IEEE 802	3	Ack
8	19.90583100(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
9	22.94008200(	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
10	24.39664600(	fe80::205:c2a:8c14:d6(	fe80::205:c2a:8c7f:6c(	ICMPv6	74	RPL Control (Destination Advertisement Object)
11	24.42080100(			IEEE 802	3	Ack
12	33.65480300(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
13	37.71684200(	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
14	38.81435100(	fe80::205:c2a:8c14:d6(	fe80::205:c2a:8c7f:6c(	ICMPv6	74	RPL Control (Destination Advertisement Object)
15	38.83927000(			IEEE 802	3	Ack
16	56.71338900(	fe80::205:c2a:8c7f:6c(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
17	60.07678700(	fe80::205:c2a:8c14:d6(	ff02::1a	ICMPv6	95	RPL Control (DODAG Information Object)
18	61.47275200(	fe80::205:c2a:8c14:d6(	fe80::205:c2a:8c7f:6c(	ICMPv6	74	RPL Control (Destination Advertisement Object)
19	61.49841500(			IEEE 802	3	Ack

Fig. 123 RPL k=10

- 1r ACK:

k=1 -> 6 msg / 28,27 seg = 0,21 missatges /seg
k=2 -> 8 msg / 32,31 seg = 0,25 missatges /seg
k=10 -> 7 msg / 16,50 seg = 0,42 missatges /seg

- 2n ACK:

k=1 -> 12 msg / 57,19 seg = 0,21 missatges /seg
k=2 -> 12 msg / 42,79 seg = 0,28 missatges /seg
k=10 -> 11 msg / 24,42 seg = 0,45 missatges /seg

- 3r ACK:

k=1 -> 16 msg / 111,47 seg = 0,14 missatges /seg
k=2 -> 16 msg / 51,82 seg = 0,31 missatges /seg
k=10 -> 15 msg / 38,84 seg = 0,39 missatges /seg

- 4t ACK:

k=1 -> 20 msg / 212,42 seg = 0,094 missatges /seg
k=2 -> 19 msg / 75,44 seg = 0,25 missatges /seg
k=10 -> 19 msg / 61,50 seg = 0,31 missatges /seg

6.9. Prova addicional: Caiguda d'un Enllaç

Tenim un escenari format pel node arrel, que compta amb dos nodes veïns amb visió directa a només un sol salt (nodes A i B), i després tenim un altre node (C) que es troba a dos salts del node arrel.

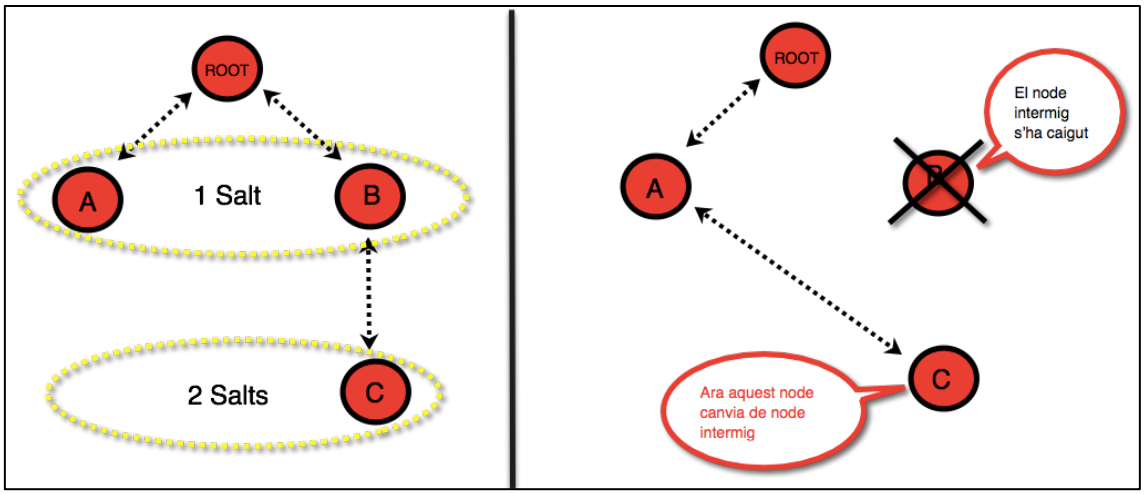


Fig. 124 Caiguda d'un enllaç

La idea és veure mitjançant la configuració de RPL, si en cas de caiguda del node B, el node C ara per arribar al node root, passi el seu tràfic a través del node A.

Neighbors

```
fe80::205:c2a:8c7f:6c08
fe80::205:c2a:8c6b:b247
```

Routes

```
aaaa::205:c2a:8c7f:6c08/128 (via fe80::205:c2a:8c7f:6c08) 16711413s
aaaa::205:c2a:8c14:d607/128 (via fe80::205:c2a:8c7f:6c08) 16711411s
aaaa::205:c2a:8c6b:b247/128 (via fe80::205:c2a:8c6b:b247) 16711411s
```

Fig. 125 Configuració inicial RPL

Aquesta proposta, no s'ha pogut implementar correctament degut a la dificultat per "refrescar" la informació de ContikiRPL front a la caiguda de nodes veïns. I tampoc es pot assegurar que la configuració de RPL del node C de la xarxa, permeti que arribi fins a node arrel a través de qualsevol dels dos nodes A o B; i en cas de fallada d'un d'aquests, pugui commutar entre els dos.

Per tant no s'ha arribat a veure, la idea era la que es descrivia en la Fig. 124; però si tombem l'enllaç de l'escenari inicial; no aconseguim commutar al segon escenari.

ANNEX H: FITXERS I SOFTWARE CONTIKI

README llibreria libmc1322x:

```
libmc1322x is a library, build system, test code, and utilities for
using the mc13224v from Freescale.

Getting Started
---------------
$ cd tests
$ make

this will build all the test files in libmc1322x/tests for each board
defined in libmc1322x/board. You will have programs like:

rftest-tx_redbee-dev.bin
rftest-tx_redbee-r1.bin

rftest-rx_redbee-dev.bin
rftest-rx_redbee-r1.bin

if you only wanted to build binaries for one board you can do:

$ make BOARD=redbee-dev

You can use mc1322x-load.pl in libmc1322x/tools to run your code:

$ ../tools/mc1322x-load.pl -f rftest-tx_redbee-dev.bin


Incorporating libmc1322x into your own code
-------------------------------------------

The best way to incorporate libmc1322x into your code is as a git
submodule:

$ mkdir newproject
$ cd newproject
$ git init

Initialized empty Git repository in /home/malvira/newproject/.git/

$ git submodule add git://git.devl.org/git/malvira/libmc1322x.git

This will add libmc1322x to your repository. Now to setup the
Makefile:

$ cp libmc1322x/tests/Makefile .

You need to edit the Makefile to point MC1322X to your libmc1322x
submodule:
```

Change line 1

MC1322X := ..

to

MC1322X := libmc1322x

and edit COBJS and TARGETS accordings. COBJS are all of your common code for any of your programs. TARGETS are the names of your programs.

For instance, you can have a common routine that prints a welcome message that is used by two programs a and b. You would add common.o to COBJS:

COBJS:= common.o

and your target line would read:

TARGETS := a b

COBJS are made for each board --- so it is ok to have board specific code in there. As an example, tests uses this in tests.c to print the name of the board in the welcome message. You could also use this to change your GPIO mappings between boards.

Rftestrx2pcap.pl (script per a fer la captura en temps real amb Wireshark):

```
#!/usr/bin/perl -w

use Device::SerialPort;
use Term::ReadKey;
use Getopt::Long;
use Time::HiRes qw(usleep gettimeofday);

use strict;

my $filename = '';
my $second = '';
my $term = '/dev/ttyUSB0';
my $baud = '115200';
my $verbose;

GetOptions (
            'terminal=s' => \$term,
            'baud=s' => \$baud,
   );

$| = 1;

# TODO: add help argument
#    print "Example usage: rftestrx2pcap.pl -t /dev/ttyS0 -b 9600\n";
#    exit;

my $ob = Device::SerialPort->new ($term) or die "Can't start $term\n";
   # next test will die at runtime unless $ob

$ob->baudrate($baud);
$ob->parity('none');
$ob->databits(8);
$ob->stopbits(1);
$ob->read_const_time(1000); # 1 second per unfulfilled "read" call

my $str = '';
my ($sec, $usec, $len);
my @frame;

my $magic = 0xa1b2c3d4;
my $major = 2;
my $minor = 4;
my $zone = 0;
my $sig = 0;
my $snaplen = 0xffff;
my $network = 230; # 802.15.4 no FCS

my $newpacket = 0;
my $len = 0;
my $file_empty = 1;

print pack('LSSLLLL',($magic,$major,$minor,$zone,$sig,$snaplen,$network));
```

```
while(1) {
   my ($count, $c) = $ob->read(1);

   if (defined($count) && ($count != 0)) {
          $str .= $c;
          # match if ends in \n or \r and process line
          if(($str =~ /\n$/) ||
            ($str =~ /\r$/)) {
             if($str =~ /^rftest-rx --- len 0x(\w\w)/) {
                    #new packet
                    ($sec, $usec) = gettimeofday;
                    $len = hex($1);
                    print STDERR "rftestline: $sec $usec $len $str";
                    $newpacket = 1;
             } elsif($str =~ /^\w+/) {
                    # dataline, write out pcap entry
                    $str =~ /(.+)/;
                    chomp $str;

          my @data = split(' ',$1);

                    # do header if this is a new packet
                    if($newpacket == 1) {
                       $newpacket = 0;
                       print pack('LLLL',($sec,$usec,$len,$len));
                       print STDERR "new packet: $sec $usec $len " . ($len) . "\n\r";
                       # This header starts the file
                       if ($file_empty == 1) {
                              $file_empty = 0;
                       }
                    }
                    # packet payload (don't start the file with a payload)
                    if ($file_empty == 0) {
                       print STDERR "dataline: ";
                       print STDERR $str . "\n\r";

                       foreach my $data (@data) {
                       print pack ('C',hex($data));
                       }
                    }

             }
             $str = '';
          }
   }
}

$ob -> close or die "Close failed: $!\n";
ReadMode 0;
undef $ob;  # closes port AND frees memory in perl
exit;
```

Fitxers de configuració:

Capa Física:
- **contiki-2.7/cpu/mc1322x/config.c**
- **contiki-2.7/cpu/mc1322x/init.c**

Capa MAC
- **contiki-2.7/platform/econotag/contiki-conf.h**

RPL
- **contiki-2.7/core/net/rpl/rpl-conf.h**
- **contiki-2.7/core/net/rpl/rpl-private.h**

config.c

```
/* MC1322x flash config system */

#include <mc1322x.h>
#include "config.h"

/* debug */
#define DEBUG DEBUG_FULL
#include "net/uip-debug.h"

mc1322xConfig mc1322x_config;

void dump_bytes(uint32_t addr, uint16_t num);

/* takes an mc1322xConf and initializes to default values */
void mc1322x_config_set_default(mc1322xConfig *c) {
        nvmType_t type;
        c->magic = MC1322X_CONFIG_MAGIC;
        c->version = MC1322X_CONFIG_VERSION;
        c->eui = 0;
   /* RF_CHANNEL = 26  */
        c->channel = 15;
        c->power = 0x01;
        c->flags.demod = DEMOD_DCD;
        c->flags.autoack = AUTOACK;
        nvm_detect(gNvmInternalInterface_c, &type);
        c->flags.nvmtype = type;
}

/* write out config to flash */
void mc1322x_config_save(mc1322xConfig *c) {
        nvmErr_t err;
        err = nvm_erase(gNvmInternalInterface_c, c->flags.nvmtype, 1 <<
MC1322X_CONFIG_PAGE/4096);
        err = nvm_write(gNvmInternalInterface_c, c->flags.nvmtype, (uint8_t *)c,
MC1322X_CONFIG_PAGE, sizeof(mc1322xConfig));
}
```

```
*/ load the config from flash to the pass conf structure */
void mc1322x_config_restore(mc1322xConfig *c) {
	nvmErr_t err;
	nvmType_t type;
	if (c->flags.nvmtype == 0) { nvm_detect(gNvmInternalInterface_c, &type); }
	c->flags.nvmtype = type;
	err = nvm_read(gNvmInternalInterface_c, c->flags.nvmtype, c,
MC1322X_CONFIG_PAGE, sizeof(mc1322xConfig));
}

/* check the flash for magic number and proper version */
int mc1322x_config_valid(mc1322xConfig *c) {
	if (c->magic == MC1322X_CONFIG_MAGIC &&
	    c->version == MC1322X_CONFIG_VERSION) {
		return 1;
	} else {
#if DEBUG
		if (c->magic != MC1322X_CONFIG_MAGIC) { PRINTF("config bad magic
%04x\n\r", c->magic); }
		if (c->version != MC1322X_CONFIG_MAGIC) { PRINTF("config bad version
%04x\n\r", c->version); }
#endif
		return -1;
	}
}

void mc1322x_config_print(void) {
	uint64_t eui64;
	PRINTF("mc1322x config:\n\r");
	PRINTF("  magic:    %04x\n\r", mc1322x_config.magic);
	PRINTF("  version:  %d\n\r",   mc1322x_config.version);
	PRINTF("  eui:      %08x%08x\n\r", (uint32_t)(mc1322x_config.eui>>32),
(uint32_t)(mc1322x_config.eui & 0xffffffff));
	PRINTF("  channel:  %d\n\r",   15);
	PRINTF("  power:    %d\n\r",   0x01);
	PRINTF("  flags:    %08x\n\r",   mc1322x_config.flags);
	PRINTF("     demod:    %d\n\r",   mc1322x_config.flags.demod);
	PRINTF("     autoack:  %d\n\r",   mc1322x_config.flags.autoack);
	PRINTF("     nvm type: %d\n\r",   mc1322x_config.flags.nvmtype);
}

void dump_bytes(uint32_t addr, uint16_t num) {
	uint32_t buf[num/4];
	nvmErr_t err;
	uint16_t i;

	err = nvm_read(gNvmInternalInterface_c, mc1322x_config.flags.nvmtype, (uint8_t *)buf,
addr, num);
	PRINTF("nvm_read returned: 0x%02x\r\n", err);

	for(i=0; i < num/4; i++) {
		printf("0x%08x\r\n", (unsigned int)buf[i]);
	}
}
```

Init.c

```
#include <stdio.h>

/* debug */
#define DEBUG DEBUG_FULL
#include "net/uip-debug.h"

/* contiki */
#include "sys/process.h"

/* mc1322x */
#include "mc1322x.h"
#include "contiki-maca.h"
#include "config.h"

/* Threshold for buck converter; buck will be disabled if vbatt is below this */
#define MC1322X_BUCK_THRES 2425
/* Hysterisis window around buck threshold */
#define MC1322X_BUCK_WINDOW 150
#define MC1322X_BUCK_THRES_H (MC1322X_BUCK_THRES +
MC1322X_BUCK_WINDOW/2)
#define MC1322X_BUCK_THRES_L (MC1322X_BUCK_THRES -
MC1322X_BUCK_WINDOW/2)
/* Time between vbatt checks for the buck */
#define MC1322X_BUCK_MONITOR_PERIOD 600 * CLOCK_SECOND

/* periodically poll adc_vbatt and manages the buck appropriately */
static struct etimer et_buck;
PROCESS(buck_monitor, "buck monitor");
PROCESS_THREAD(buck_monitor, ev, data)
{

        PROCESS_BEGIN();
        PRINTF("starting vbatt monitor\n");

        etimer_set(&et_buck, MC1322X_BUCK_MONITOR_PERIOD);

        while (1) {
                PROCESS_WAIT_EVENT();
                if(etimer_expired(&et_buck))
                {
                        adc_service();
                        PRINTF("buck monitor: vbatt: %d mV\n\r", adc_vbatt);
                        if( CRM->VREG_CNTLbits.BUCK_EN == 1 && adc_vbatt <
MC1322X_BUCK_THRES_L ) {
                                PRINTF("vbatt low, disabling buck\n\r", adc_vbatt);
                                CRM->SYS_CNTLbits.PWR_SOURCE = 0;
                                CRM->VREG_CNTLbits.BUCK_SYNC_REC_EN = 0;
                CRM->VREG_CNTLbits.BUCK_BYPASS_EN = 1;
                                        CRM->VREG_CNTLbits.BUCK_EN = 0;
                                } else if ( CRM->VREG_CNTLbits.BUCK_EN == 0 &&
        adc_vbatt > MC1322X_BUCK_THRES_H ) {
```

```
                        PRINTF("vbatt high, enabling buck\n\r", adc_vbatt);
                                CRM->SYS_CNTLbits.PWR_SOURCE = 1;
                                CRM->VREG_CNTLbits.BUCK_SYNC_REC_EN = 1;
                                CRM->VREG_CNTLbits.BUCK_BYPASS_EN = 0;
                                CRM->VREG_CNTLbits.BUCK_EN = 1;
                        }
                        etimer_set(&et_buck, MC1322X_BUCK_MONITOR_PERIOD);
                }
        }

        PROCESS_END();

}

void buck_setup(void) {
        nvmType_t type;
        nvmErr_t err;
        volatile int i;

        default_vreg_init();

        while(CRM->STATUSbits.VREG_1P5V_RDY == 0) { continue; }
        while(CRM->STATUSbits.VREG_1P8V_RDY == 0) { continue; }

        /* takes time for the flash supply to fail (if there is no buck) */
        /* spin while this happens doing nvm_detects */
        /* XXX todo: don't probe buck if Vbatt < 2.5V */

        adc_service();
        PRINTF("vbatt: %04u mV\n\r", adc_vbatt);

        type = 1;
        for(i = 0; i < 128 && type != 0; i++) {
                err = nvm_detect(gNvmInternalInterface_c, &type);
        }
        if (type == gNvmType_NoNvm_c)
        {
                PRINTF("NVM failed without buck, trying with buck\n\r");

                if (adc_vbatt < MC1322X_BUCK_THRES_L)
                {
                        PRINTF("Vbatt is low, bypassing buck\n\r");
                        CRM->SYS_CNTLbits.PWR_SOURCE = 0;
                        CRM->VREG_CNTLbits.BUCK_SYNC_REC_EN = 0;
                        CRM->VREG_CNTLbits.BUCK_BYPASS_EN = 1;
                        CRM->VREG_CNTLbits.BUCK_EN = 0;
                } else {
                        CRM->SYS_CNTLbits.PWR_SOURCE = 1;
                        CRM->VREG_CNTLbits.BUCK_SYNC_REC_EN = 1;
                        CRM->VREG_CNTLbits.BUCK_BYPASS_EN = 0;
                        CRM->VREG_CNTLbits.BUCK_EN = 1;
                }
        while(CRM->STATUSbits.VREG_BUCK_RDY == 0) { continue; }
                        CRM->VREG_CNTLbits.VREG_1P5V_SEL = 3;
                        CRM->VREG_CNTLbits.VREG_1P5V_EN = 3;
                        CRM->VREG_CNTLbits.VREG_1P8V_EN = 1;
```

```
	while(CRM->STATUSbits.VREG_1P5V_RDY == 0) { continue; }
			while(CRM->STATUSbits.VREG_1P8V_RDY == 0) { continue; }

			type = 1;
			for(i = 0; i < 128 && type != 0; i++) {
				err = nvm_detect(gNvmInternalInterface_c, &type);

		}
		if (type != gNvmType_NoNvm_c) {
			PRINTF("buck ok\n\r");
	/* start a process to monitor vbatt and enable/disable the buck as necessary */
			process_start(&buck_monitor, NULL);
		} else {
			printf("fatal: couldn't detect NVM\n\r");
		}
	} else {
		PRINTF("NVM ok without buck\n\r");
	}
}

/* setup the RTC */
/* try to start the 32kHz xtal */
void rtc_setup(void) {
	volatile uint32_t rtc_count;
	volatile uint32_t i;

	ring_osc_off();
	xtal32_on();
	xtal32_exists();
	rtc_count = CRM->RTC_COUNT;
	PRINTF("trying to start 32kHz xtal\n\r");

	for(i = 0; i < 150000 && CRM->RTC_COUNT == rtc_count; i++) { continue; }
	if(CRM->RTC_COUNT == rtc_count) {
		PRINTF("32xtal failed, using ring osc\n\r");
		CRM->SYS_CNTLbits.XTAL32_EXISTS = 0;
		CRM->XTAL32_CNTLbits.XTAL32_EN = 0;
		ring_osc_on();

		/* Set default tune values from datasheet */
		CRM->RINGOSC_CNTLbits.ROSC_CTUNE = 0x6;
		CRM->RINGOSC_CNTLbits.ROSC_FTUNE = 0x17;

		/* Trigger calibration */
		rtc_calibrate();
		PRINTF("RTC calibrated to %d Hz\r\n", rtc_freq);
	} else {
		PRINTF("32kHz xtal started\n\r");
		rtc_freq = 32768;
	}
}
```

```
/* call mc1322x_init once to initalize everything with the current config */
void mc1322x_init(void) {

	/* XXX TODO load config from flash */
	/* config should say what uart to use for debug console */
	/* config should also set the baud rate */
	/* for now, just clean up contiki-conf.h */
	/* maybe factor into conf.h -> contiki-conf.h and mc1322x-conf.h platform-conf.h */

	/* print out config in debug */
	/* initialize the uarts */
	uart_init(CONSOLE_UART, CONSOLE_BAUD);
	PRINTF("mc1322x init\n\r");

	adc_init();
	ctimer_init();
	process_init();
	process_start(&etimer_process, NULL);
	process_start(&contiki_maca_process, NULL);
	buck_setup();

	/* start with a default config */

	mc1322x_config_restore(&mc1322x_config);
	if ( mc1322x_config_valid(&mc1322x_config) != 1 ) {
		PRINTF("flash invalid\n\r");
		/* save the default config to flash */
		mc1322x_config_set_default(&mc1322x_config);
		mc1322x_config_save(&mc1322x_config);
	}

#if DEBUG_FULL
	mc1322x_config_print();
#endif

	/* setup the radio */
	maca_init();
//	set_power(mc1322x_config.power);
	set_power(0x01);
	set_channel(15);
	set_demodulator_type(mc1322x_config.flags.demod);
	set_prm_mode(mc1322x_config.flags.autoack);

	/* must be done AFTER maca_init */
	/* the radio calibration appears to clobber the RTC trim caps */
	rtc_setup();
	rtimer_init();
	clock_init();

}
```

contiki-conf.h

```
#ifndef __CONTIKI_CONF_H__
#define __CONTIKI_CONF_H__

#include <stdint.h>

/* mc1322x files */
#include "contiki-mc1322x-conf.h"

/* Econotag I tune parameters */
#define ECONOTAG_CTUNE_4PF 1
/* Coarse tune: add 0-15 pf (CTUNE is 4 bits) */
#define ECONOTAG_CTUNE 11
/* Fine tune: add FTUNE * 156fF (FTUNE is 5bits) */
#define ECONOTAG_FTUNE 7

/* M12 tune parameters */
#define M12_CTUNE_4PF 1
#define M12_CTUNE 3
#define M12_FTUNE 3

/* the econotag platform will correctly detect an Econotag I (no M12) vs. Econotag II
(w/M12) */
/* and trim the main crystal accordingly */
/* this detection will be incorrect if you populate the 32.768kHz crystal on the Econotag I */
/* In that case, you should FORCE_ECONOTAG_I below */
#define FORCE_ECONOTAG_I 0

/* if you define a serial number then it will be used to comput the mac address */
/* otherwise, a random mac address in the Redwire development IAB will be used */
/* #define M12_CONF_SERIAL 0x000000 */

/* Clock ticks per second */
#define CLOCK_CONF_SECOND 100

#define CCIF
#define CLIF

#define CONSOLE_UART UART1
#define CONSOLE_BAUD 115200

#define dbg_putchar(x) uart1_putc(x)

#define USE_FORMATTED_STDIO        1
#define MACA_DEBUG                 0
#define CONTIKI_MACA_RAW_MODE      0

#define BLOCKING_TX 1
// auto ack
#define MACA_AUTOACK 1
#define NULLRDC_CONF_802154_AUTOACK_HW 1
#define USE_WDT 0

#ifndef WDT_TIMEOUT
#define WDT_TIMEOUT 5000 /* watchdog timeout in ms */
#endif
```

```
/* end of mc1322x specific config. */

/* start of conitki config. */
#define PLATFORM_HAS_LEDS 1
#define PLATFORM_HAS_BUTTON 1

#define RIMEADDR_CONF_SIZE              8

#if WITH_UIP6
/* Network setup for IPv6 */
#define NETSTACK_CONF_NETWORK sicslowpan_driver

// MAC mode: csma_driver = use csma to access channel ; nullmac_driver = transmit without
listening
// accés al medi
#undef NETSTACK_CONF_MAC
#define NETSTACK_CONF_MAC     nullmac_driver

// RDC MODE: nullrdc_driver = radio permanently on ; contikimac_driver = contikimac-RDC
// duty cycling
#undef NETSTACK_CONF_RDC
#define NETSTACK_CONF_RDC     nullrdc_driver

#define NETSTACK_CONF_RADIO   contiki_maca_driver
#define NETSTACK_CONF_FRAMER  framer_802154

#define NETSTACK_CONF_RDC_CHANNEL_CHECK_RATE      8
#define RIME_CONF_NO_POLITE_ANNOUCEMENTS 0
#define CXMAC_CONF_ANNOUNCEMENTS        0
#define XMAC_CONF_ANNOUNCEMENTS         0

#else /* WITH_UIP6 */
/* Network setup for non-IPv6 (rime). */

#define NETSTACK_CONF_NETWORK rime_driver
#define NETSTACK_CONF_MAC     csma_driver
#define NETSTACK_CONF_RDC     sicslowmac_driver
#define NETSTACK_CONF_RADIO   contiki_maca_driver
#define NETSTACK_CONF_FRAMER  framer_802154

#define NETSTACK_CONF_RDC_CHANNEL_CHECK_RATE      8

#define COLLECT_CONF_ANNOUNCEMENTS      1
#define RIME_CONF_NO_POLITE_ANNOUCEMENTS 0
#define CXMAC_CONF_ANNOUNCEMENTS        0
#define XMAC_CONF_ANNOUNCEMENTS         0
#define CONTIKIMAC_CONF_ANNOUNCEMENTS    0

#define CONTIKIMAC_CONF_COMPOWER        0
#define XMAC_CONF_COMPOWER              0
#define CXMAC_CONF_COMPOWER             0
#define COLLECT_NBR_TABLE_CONF_MAX_NEIGHBORS      32

#endif /* WITH_UIP6 */

#define QUEUEBUF_CONF_NUM          16

#define PACKETBUF_CONF_ATTRS_INLINE 1
```

```
#ifndef RF_CHANNEL
//canal radio
#undef RF_CHANNEL
#define RF_CHANNEL              26
#endif /* RF_CHANNEL */


#define CONTIKIMAC_CONF_BROADCAST_RATE_LIMIT 0

//pan id
#define IEEE802154_CONF_PANID       0xABCD

#define PROFILE_CONF_ON 0
#define ENERGEST_CONF_ON 0

#define AODV_COMPLIANCE
#define AODV_NUM_RT_ENTRIES 32

#define WITH_ASCII 1

#define PROCESS_CONF_NUMEVENTS 8
#define PROCESS_CONF_STATS 1

#ifdef WITH_UIP6

#define RIMEADDR_CONF_SIZE            8

#define UIP_CONF_LL_802154            1
#define UIP_CONF_LLH_LEN              0

#ifndef UIP_CONF_ROUTER
#define UIP_CONF_ROUTER               1
#endif

#ifndef UIP_CONF_IPV6_RPL
#define UIP_CONF_IPV6_RPL             1
#endif

#define NBR_TABLE_CONF_MAX_NEIGHBORS     30
#define UIP_CONF_MAX_ROUTES   30

#define UIP_CONF_ND6_SEND_RA              0
#define UIP_CONF_ND6_REACHABLE_TIME     600000
#define UIP_CONF_ND6_RETRANS_TIMER      10000

#define UIP_CONF_IPV6                 1
#define UIP_CONF_IPV6_QUEUE_PKT       0
#define UIP_CONF_IPV6_CHECKS          1

#define UIP_CONF_IPV6_REASSEMBLY      0
#define UIP_CONF_NETIF_MAX_ADDRESSES    3
#define UIP_CONF_ND6_MAX_PREFIXES       3
#define UIP_CONF_ND6_MAX_DEFROUTERS     2
#define UIP_CONF_IP_FORWARD           0
#define UIP_CONF_BUFFER_SIZE              1300
#define SICSLOWPAN_CONF_FRAG          1
#define SICSLOWPAN_CONF_MAXAGE        8
```

```
#define SICSLOWPAN_CONF_COMPRESSION_IPV6       0
#define SICSLOWPAN_CONF_COMPRESSION_HC1        1
#define SICSLOWPAN_CONF_COMPRESSION_HC01       2
#define SICSLOWPAN_CONF_COMPRESSION            SICSLOWPAN_COMPRESSION_HC06
#ifndef SICSLOWPAN_CONF_FRAG
#define SICSLOWPAN_CONF_FRAG                   1
#define SICSLOWPAN_CONF_MAXAGE                 8
#endif /* SICSLOWPAN_CONF_FRAG */
#define SICSLOWPAN_CONF_CONVENTIONAL_MAC       1
#define SICSLOWPAN_CONF_MAX_ADDR_CONTEXTS      2
#else /* WITH_UIP6 */
#define UIP_CONF_IP_FORWARD     1
#define UIP_CONF_BUFFER_SIZE    1300
#endif /* WITH_UIP6 */

#define UIP_CONF_ICMP_DEST_UNREACH 1

#define UIP_CONF_DHCP_LIGHT
#define UIP_CONF_LLH_LEN        0
#define UIP_CONF_RECEIVE_WINDOW  48
#define UIP_CONF_TCP_MSS        48
#define UIP_CONF_MAX_CONNECTIONS 4
#define UIP_CONF_MAX_LISTENPORTS 8
#define UIP_CONF_UDP_CONNS      12
#define UIP_CONF_FWCACHE_SIZE   30
#define UIP_CONF_BROADCAST      1
#define UIP_ARCH_IPCHKSUM       1
#define UIP_CONF_UDP            1
#define UIP_CONF_UDP_CHECKSUMS  1
#define UIP_CONF_PINGADDRCONF   0
#define UIP_CONF_LOGGING        0

#define UIP_CONF_TCP_SPLIT      0

/* include the project config */
/* PROJECT_CONF_H might be defined in the project Makefile */
#ifdef PROJECT_CONF_H
#include PROJECT_CONF_H
#endif /* PROJECT_CONF_H */

#endif /* __CONTIKI_CONF_H__ */
```

rpl-conf.h

```
#ifndef RPL_CONF_H
#define RPL_CONF_H

#include "contiki-conf.h"

/* Set to 1 to enable RPL statistics */
#ifndef RPL_CONF_STATS
#define RPL_CONF_STATS 0
#endif /* RPL_CONF_STATS */

/*
 * Select routing metric supported at runtime. This must be a valid
 * DAG Metric Container Object Type (see below). Currently, we only
 * support RPL_DAG_MC_ETX and RPL_DAG_MC_ENERGY.
 * When MRHOF (RFC6719) is used with ETX, no metric container must
 * be used; instead the rank carries ETX directly.
 */
#ifdef RPL_CONF_DAG_MC
#define RPL_DAG_MC RPL_CONF_DAG_MC
#else
#define RPL_DAG_MC RPL_DAG_MC_NONE
#endif /* RPL_CONF_DAG_MC */

/*
 * The objective function used by RPL is configurable through the
 * RPL_CONF_OF parameter. This should be defined to be the name of an
 * rpl_of object linked into the system image, e.g., rpl_of0.
 */
#ifdef RPL_CONF_OF
#define RPL_OF RPL_CONF_OF
#else
/* ETX is the default objective function. */
#define RPL_OF rpl_mrhof
#endif /* RPL_CONF_OF */

/* This value decides which DAG instance we should participate in by default. */
#ifdef RPL_CONF_DEFAULT_INSTANCE
#define RPL_DEFAULT_INSTANCE RPL_CONF_DEFAULT_INSTANCE
#else
#define RPL_DEFAULT_INSTANCE           0x1e
#endif /* RPL_CONF_DEFAULT_INSTANCE */

/*
 * This value decides if this node must stay as a leaf or not
 * as allowed by draft-ietf-roll-rpl-19#section-8.5
 */
```

```
#ifdef RPL_CONF_LEAF_ONLY
#define RPL_LEAF_ONLY RPL_CONF_LEAF_ONLY
#else
#define RPL_LEAF_ONLY 0
#endif

/*
 * Maximum of concurent RPL instances.
 */
#ifdef RPL_CONF_MAX_INSTANCES
#define RPL_MAX_INSTANCES     RPL_CONF_MAX_INSTANCES
#else
#define RPL_MAX_INSTANCES     1
#endif /* RPL_CONF_MAX_INSTANCES */

/*
 * Maximum number of DAGs within an instance.
 */
#ifdef RPL_CONF_MAX_DAG_PER_INSTANCE
#define RPL_MAX_DAG_PER_INSTANCE     RPL_CONF_MAX_DAG_PER_INSTANCE
#else
#define RPL_MAX_DAG_PER_INSTANCE     2
#endif /* RPL_CONF_MAX_DAG_PER_INSTANCE */

/*
 *
 */
#ifndef RPL_CONF_DAO_SPECIFY_DAG
  #if RPL_MAX_DAG_PER_INSTANCE > 1
    #define RPL_DAO_SPECIFY_DAG 1
  #else
    #define RPL_DAO_SPECIFY_DAG 0
  #endif /* RPL_MAX_DAG_PER_INSTANCE > 1 */
#else
  #define RPL_DAO_SPECIFY_DAG RPL_CONF_DAO_SPECIFY_DAG
#endif /* RPL_CONF_DAO_SPECIFY_DAG */

/*
 * The DIO interval (n) represents 2^n ms.
```

Rpl-private.h

```
#ifndef RPL_PRIVATE_H
#define RPL_PRIVATE_H

#include "net/rpl/rpl.h"

#include "lib/list.h"
#include "net/uip.h"
#include "sys/clock.h"
#include "sys/ctimer.h"
#include "net/uip-ds6.h"

/*---------------------------------------------------------------------------*/
/** \brief Is IPv6 address addr the link-local, all-RPL-nodes
    multicast address? */
#define uip_is_addr_linklocal_rplnodes_mcast(addr)         \
  ((addr)->u8[0] == 0xff) &&                               \
  ((addr)->u8[1] == 0x02) &&                               \
  ((addr)->u16[1] == 0) &&                                 \
  ((addr)->u16[2] == 0) &&                                 \
  ((addr)->u16[3] == 0) &&                                 \
  ((addr)->u16[4] == 0) &&                                 \
  ((addr)->u16[5] == 0) &&                                 \
  ((addr)->u16[6] == 0) &&                                 \
  ((addr)->u8[14] == 0) &&                                 \
  ((addr)->u8[15] == 0x1a))

/** \brief Set IP address addr to the link-local, all-rpl-nodes
    multicast address. */
#define uip_create_linklocal_rplnodes_mcast(addr)       \
  uip_ip6addr((addr), 0xff02, 0, 0, 0, 0, 0, 0, 0x001a)
/*---------------------------------------------------------------------------*/
/* RPL message types */
#define RPL_CODE_DIS                0x00   /* DAG Information Solicitation */
#define RPL_CODE_DIO                0x01   /* DAG Information Option */
#define RPL_CODE_DAO                0x02   /* Destination Advertisement Option */
#define RPL_CODE_DAO_ACK            0x03   /* DAO acknowledgment */
#define RPL_CODE_SEC_DIS            0x80   /* Secure DIS */
#define RPL_CODE_SEC_DIO            0x81   /* Secure DIO */
#define RPL_CODE_SEC_DAO            0x82   /* Secure DAO */
#define RPL_CODE_SEC_DAO_ACK        0x83   /* Secure DAO ACK */

/* RPL control message options. */
#define RPL_OPTION_PAD1               0
#define RPL_OPTION_PADN               1
#define RPL_OPTION_DAG_METRIC_CONTAINER  2
#define RPL_OPTION_ROUTE_INFO         3
#define RPL_OPTION_DAG_CONF           4
#define RPL_OPTION_TARGET             5
#define RPL_OPTION_TRANSIT            6
#define RPL_OPTION_SOLICITED_INFO     7
#define RPL_OPTION_PREFIX_INFO        8
#define RPL_OPTION_TARGET_DESC        9
#define RPL_DAO_K_FLAG              0x80 /* DAO ACK requested */
#define RPL_DAO_D_FLAG              0x40 /* DODAG ID present */
/*---------------------------------------------------------------------------*/
```

```
/* RPL IPv6 extension header option. */
#define RPL_HDR_OPT_LEN                 4
#define RPL_HOP_BY_HOP_LEN              (RPL_HDR_OPT_LEN + 2 + 2)
#define RPL_HDR_OPT_DOWN                0x80
#define RPL_HDR_OPT_DOWN_SHIFT          7
#define RPL_HDR_OPT_RANK_ERR            0x40
#define RPL_HDR_OPT_RANK_ERR_SHIFT      6
#define RPL_HDR_OPT_FWD_ERR             0x20
#define RPL_HDR_OPT_FWD_ERR_SHIFT       5
/*---------------------------------------------------------------------------*/
/* Default values for RPL constants and variables. */

/* The default value for the DAO timer. */
#ifdef RPL_CONF_DAO_LATENCY
#define RPL_DAO_LATENCY             RPL_CONF_DAO_LATENCY
#else /* RPL_CONF_DAO_LATENCY */
#define RPL_DAO_LATENCY             (CLOCK_SECOND * 4)
#endif /* RPL_DAO_LATENCY */

/* Special value indicating immediate removal. */
#define RPL_ZERO_LIFETIME           0

#define RPL_LIFETIME(instance, lifetime) \
        ((unsigned long)(instance)->lifetime_unit * (lifetime))

#ifndef RPL_CONF_MIN_HOPRANKINC
#define RPL_MIN_HOPRANKINC          256
#else
#define RPL_MIN_HOPRANKINC          RPL_CONF_MIN_HOPRANKINC
#endif
#define RPL_MAX_RANKINC             (7 * RPL_MIN_HOPRANKINC)

#define DAG_RANK(fixpt_rank, instance) \
  ((fixpt_rank) / (instance)->min_hoprankinc)

/* Rank of a virtual root node that coordinates DAG root nodes. */
#define BASE_RANK                   0

/* Rank of a root node. */
#define ROOT_RANK(instance)         (instance)->min_hoprankinc

#define INFINITE_RANK               0xffff

/* Represents 2^n ms. */
/* Default value according to the specification is 3 which
   means 8 milliseconds, but that is an unreasonable value if
   using power-saving / duty-cycling    */
#ifdef RPL_CONF_DIO_INTERVAL_MIN
#define RPL_DIO_INTERVAL_MIN        RPL_CONF_DIO_INTERVAL_MIN
#else
#define RPL_DIO_INTERVAL_MIN        12
#endif
/* Maximum amount of timer doublings. */
#ifdef RPL_CONF_DIO_INTERVAL_DOUBLINGS
#define RPL_DIO_INTERVAL_DOUBLINGS  RPL_CONF_DIO_INTERVAL_DOUBLINGS
#else
#define RPL_DIO_INTERVAL_DOUBLINGS  8
#endif
```

```
/* Default DIO redundancy. */
#ifdef RPL_CONF_DIO_REDUNDANCY
#define RPL_DIO_REDUNDANCY          RPL_CONF_DIO_REDUNDANCY
#else
#define RPL_DIO_REDUNDANCY          15
#endif

/* Expire DAOs from neighbors that do not respond in this time. (seconds) */
#define DAO_EXPIRATION_TIMEOUT          60
/*---------------------------------------------------------------------------*/
#define RPL_INSTANCE_LOCAL_FLAG        0x80
#define RPL_INSTANCE_D_FLAG            0x40

/* Values that tell where a route came from. */
#define RPL_ROUTE_FROM_INTERNAL        0
#define RPL_ROUTE_FROM_UNICAST_DAO     1
#define RPL_ROUTE_FROM_MULTICAST_DAO    2
#define RPL_ROUTE_FROM_DIO             3

/* DAG Mode of Operation */
#define RPL_MOP_NO_DOWNWARD_ROUTES      0
#define RPL_MOP_NON_STORING             1
#define RPL_MOP_STORING_NO_MULTICAST    2
#define RPL_MOP_STORING_MULTICAST       3

#ifdef  RPL_CONF_MOP
#define RPL_MOP_DEFAULT                 RPL_CONF_MOP
#else
#define RPL_MOP_DEFAULT                 RPL_MOP_STORING_NO_MULTICAST
#endif

/*
 * The ETX in the metric container is expressed as a fixed-point value
 * whose integer part can be obtained by dividing the value by
 * RPL_DAG_MC_ETX_DIVISOR.
 */
#define RPL_DAG_MC_ETX_DIVISOR              128

/* DIS related */
#define RPL_DIS_SEND                    1
#ifdef  RPL_DIS_INTERVAL_CONF
#define RPL_DIS_INTERVAL                RPL_DIS_INTERVAL_CONF
#else
#define RPL_DIS_INTERVAL                60
#endif
#define RPL_DIS_START_DELAY             5
/*---------------------------------------------------------------------------*/
/* Lollipop counters */

#define RPL_LOLLIPOP_MAX_VALUE          255
#define RPL_LOLLIPOP_CIRCULAR_REGION     127
#define RPL_LOLLIPOP_SEQUENCE_WINDOWS    16
#define RPL_LOLLIPOP_INIT               (RPL_LOLLIPOP_MAX_VALUE -
RPL_LOLLIPOP_SEQUENCE_WINDOWS + 1)
#define RPL_LOLLIPOP_INCREMENT(counter)                         \
  do {                                                          \
    if((counter) > RPL_LOLLIPOP_CIRCULAR_REGION) {              \
      (counter) = ((counter) + 1) & RPL_LOLLIPOP_MAX_VALUE;     \
    } else {                                                    \
      (counter) = ((counter) + 1) & RPL_LOLLIPOP_CIRCULAR_REGION;   \
    }                                                           \
  } while(0)
```

```
#define RPL_LOLLIPOP_IS_INIT(counter)                    \
  ((counter) > RPL_LOLLIPOP_CIRCULAR_REGION)
/*---------------------------------------------------------------------------*/
/* Logical representation of a DAG Information Object (DIO.) */
struct rpl_dio {
  uip_ipaddr_t dag_id;
  rpl_ocp_t ocp;
  rpl_rank_t rank;
  uint8_t grounded;
  uint8_t mop;
  uint8_t preference;
  uint8_t version;
  uint8_t instance_id;
  uint8_t dtsn;
  uint8_t dag_intdoubl;
  uint8_t dag_intmin;
  uint8_t dag_redund;
  uint8_t default_lifetime;
  uint16_t lifetime_unit;
  rpl_rank_t dag_max_rankinc;
  rpl_rank_t dag_min_hoprankinc;
  rpl_prefix_t destination_prefix;
  rpl_prefix_t prefix_info;
  struct rpl_metric_container mc;
};
typedef struct rpl_dio rpl_dio_t;

#if RPL_CONF_STATS
/* Statistics for fault management. */
struct rpl_stats {
  uint16_t mem_overflows;
  uint16_t local_repairs;
  uint16_t global_repairs;
  uint16_t malformed_msgs;
  uint16_t resets;
  uint16_t parent_switch;
};
typedef struct rpl_stats rpl_stats_t;

extern rpl_stats_t rpl_stats;
#endif
/*---------------------------------------------------------------------------*/
/* RPL macros. */

#if RPL_CONF_STATS
#define RPL_STAT(code)          (code)
#else
#define RPL_STAT(code)
#endif /* RPL_CONF_STATS */
/*---------------------------------------------------------------------------*/
/* Instances */
extern rpl_instance_t instance_table[];
extern rpl_instance_t *default_instance;
```

```
/* ICMPv6 functions for RPL. */
void dis_output(uip_ipaddr_t *addr);
void dio_output(rpl_instance_t *, uip_ipaddr_t *uc_addr);
void dao_output(rpl_parent_t *, uint8_t lifetime);
void dao_output_target(rpl_parent_t *, uip_ipaddr_t *, uint8_t lifetime);
void dao_ack_output(rpl_instance_t *, uip_ipaddr_t *, uint8_t);

/* RPL logic functions. */
void rpl_join_dag(uip_ipaddr_t *from, rpl_dio_t *dio);
void rpl_join_instance(uip_ipaddr_t *from, rpl_dio_t *dio);
void rpl_local_repair(rpl_instance_t *instance);
void rpl_process_dio(uip_ipaddr_t *, rpl_dio_t *);
int rpl_process_parent_event(rpl_instance_t *, rpl_parent_t *);

/* DAG object management. */
rpl_dag_t *rpl_alloc_dag(uint8_t, uip_ipaddr_t *);
rpl_instance_t *rpl_alloc_instance(uint8_t);
void rpl_free_dag(rpl_dag_t *);
void rpl_free_instance(rpl_instance_t *);

/* DAG parent management function. */
rpl_parent_t *rpl_add_parent(rpl_dag_t *, rpl_dio_t *dio, uip_ipaddr_t *);
rpl_parent_t *rpl_find_parent(rpl_dag_t *, uip_ipaddr_t *);
rpl_parent_t *rpl_find_parent_any_dag(rpl_instance_t *instance, uip_ipaddr_t *addr);
void rpl_nullify_parent(rpl_parent_t *);
void rpl_remove_parent(rpl_parent_t *);
void rpl_move_parent(rpl_dag_t *dag_src, rpl_dag_t *dag_dst, rpl_parent_t *parent);
rpl_parent_t *rpl_select_parent(rpl_dag_t *dag);
rpl_dag_t *rpl_select_dag(rpl_instance_t *instance,rpl_parent_t *parent);
void rpl_recalculate_ranks(void);

/* RPL routing table functions. */
void rpl_remove_routes(rpl_dag_t *dag);
void rpl_remove_routes_by_nexthop(uip_ipaddr_t *nexthop, rpl_dag_t *dag);
uip_ds6_route_t *rpl_add_route(rpl_dag_t *dag, uip_ipaddr_t *prefix,
                               int prefix_len, uip_ipaddr_t *next_hop);
void rpl_purge_routes(void);

/* Objective function. */
rpl_of_t *rpl_find_of(rpl_ocp_t);

/* Timer functions. */
void rpl_schedule_dao(rpl_instance_t *);
void rpl_reset_dio_timer(rpl_instance_t *);
void rpl_reset_periodic_timer(void);

/* Route poisoning. */
void rpl_poison_routes(rpl_dag_t *, rpl_parent_t *);

#endif /* RPL_PRIVATE_H */
```

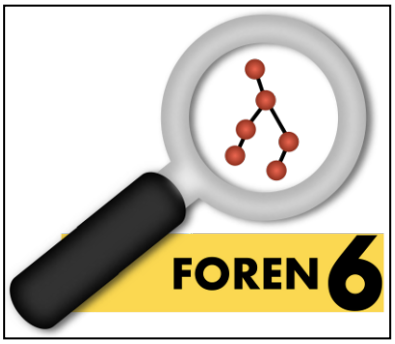
ANNEX I: SOBRE CASOS DE FUTUR I INVESTIGACIÓ

En aquest annex, sobre els casos de futur i investigacions que es poden seguir fent en aquest àmbit, destacaré una eina que és força interessant, per a analitzar el trànsit 6LoWPAN, i per a fer representacions visuals de les xarxes que formem, i que seria útil també de cara a futures pràctiques de laboratori.

També, en el segon apartat, parlaré de notícies que estan sortint actualment sobre nous Sistemes Operatius que pensats per a la internet de les coses i on es veu clarament l'aposta de les grans empreses conegudes a nivell mundial com Google, que hi estan dedicant esforços i inversions en aquest àmbit.

- **Eina Foren6**

Foren6 és una eina d'anàlisi, de xarxes 6LoWPAN no intrusiva.
Aprofita dispositius sniffers passius per reconstruir una representació visual i textual de la informació de la xarxa per donar suport a Internet en el món real de les aplicacions de les coses on altres mitjans de depuració (per cable o un control basat en xarxa) són massa costosos o poc pràctic.
Permet visualitzar la nostra xarxa 6LoWPAN: fa servir sniffers per capturar el tràfic 6LoWPAN i ens renderitza l'estat de la xarxa mitjançant una interfície gràfica d'usuari.
Permet també detectar problemes de routing: El protocol RPL, és un estàndard IETF emergent. Foren6 captura tota la informació relacionada amb RPL i identifica conductes anormals.
Diagnòstic in situ. Foren6 captura paquets en directe de les xarxes desplegades de manera no intrusiva. Múltiples sniffers es poden combinar per a una cobertura estesa.
Depuració orientada. Es pot rebobinar la captura de paquets, reproduir una traça paquet anterior i navegar a través de diferents superposicions per identificar problemes.
Personalitza la teva infraestructura. El visor de xarxa utilitza posicions flotants, o dissenys definits per l'usuari per visualitzar els sensors en el seu entorn real.
Ofereix suport d'Android: s'està desenvolupant un port d'Android, que permetrà visualitzar per exemple la xarxa 6LoWPAN en una tablet.
És lliure i de Codi Obert: No té cap cost, a més es poden seguir les actualitzacions de desenvolupament de codi obert; hi ha també una llista de correu, foren6-dev: http://lists.cetic.be/cgi-bin/mailman/listinfo/foren6-dev



Instal·lació
http://cetic.github.io/foren6/install.html
Prova d'exemple:
http://cetic.github.io/foren6/example2.html

- Fent de Sniffer amb un Econotag amb Foren6

- Necessitem libmc1322x, i compilar els firmwares

```
git clone https://github.com/malvira/libmc1322x.git
cd libmc1322x/tests
make
```

- Programem un Econotag amb el programa rftest-rx (sniffer).
- Creem un fitxer fifo amb el terminal:

```
mkfifo /path/to/fifo.pcap
```

- Per executar Foren6: entrem al path on el tenim ubicat i feim 'make run':
  - *cd foren6/*
  - *make run*

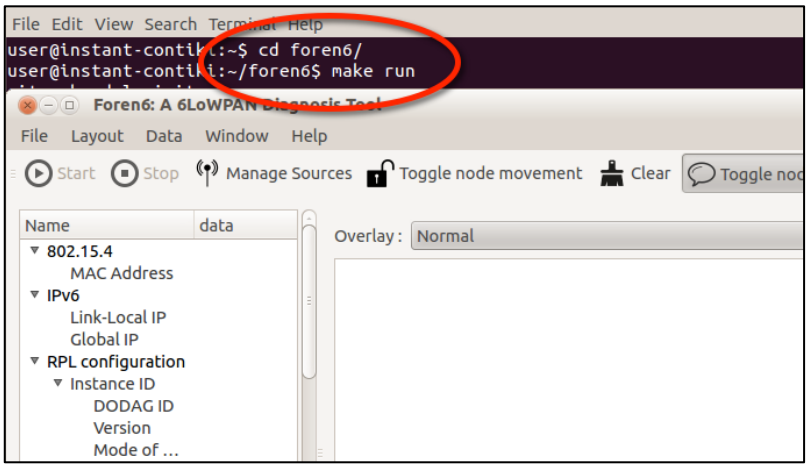


Fig. 126 Executar Foren6

- Establim el fitxer creat abans "/path/to/fifo.pcap" com a "source".
- Manage Sources —> Busquem fifo.pcap, establim el PCAP type.

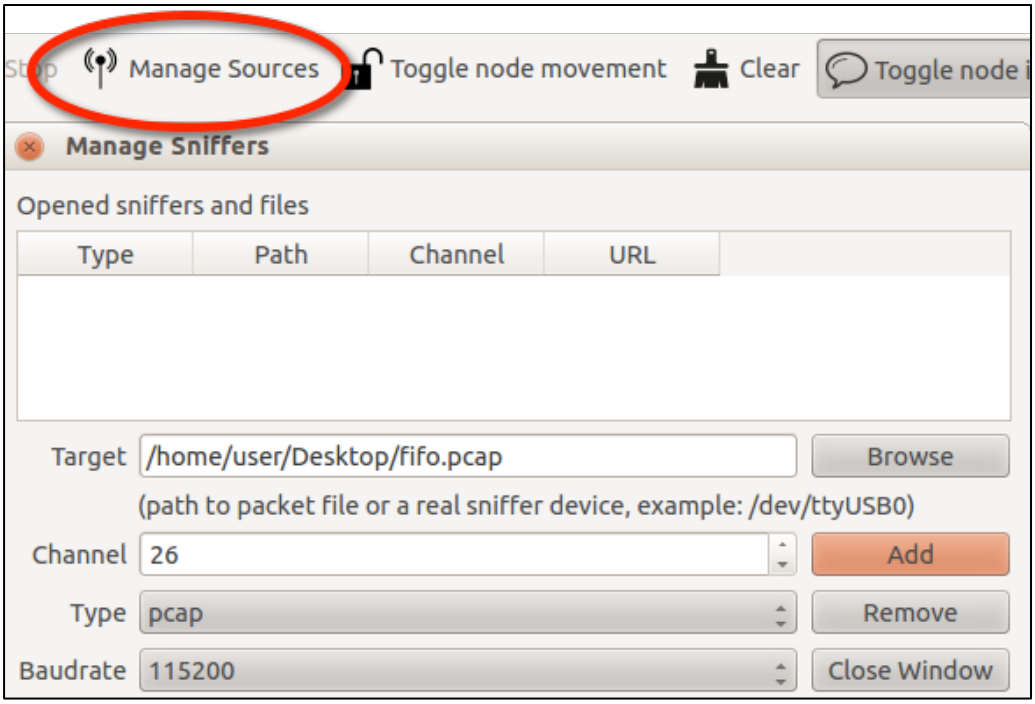


Fig. 127 Manage Sources Foren6

- En una consola obrirem el rftestrx2pcap.py de la llibreria libmc1322x:

```
cd ..
./tests/rftestrx2pcap.py /dev/ttyUSB1 24 /path/to/fifo.pcap
```

- Tornem al programa Foren6 i clicarem el Boto Start.

- **Projecte: Brillo O.S.**

En aquest Projecte nosaltres hem fet servir Contiki O.S. que és un sistema operatiu per a la internet de les coses.
Veiem com hi ha "grans" empreses com Google que estan invertint i investigant també en aquest sentit que tindrà molta d'importància en un futur. Una d'aquestes és Google, que en el seu últim congrés: el Google I/O 2015. Ens va presentar el que serà seu nou projecte de Sistema Operatiu per a la Internet de les Coses.

Project Brillo, l'han anomenat, i serà un nou S.O. presentat per Google, del que podrem veure'n la seva primera implementació (developer Preview) abans de final d'aquest any.
Es tracta d'un sistema operatiu derivat d'Android, però minimitzat per poder ser utilitzat en dispositius amb poca potència. Suportarà connexions WiFi, Bluetooth LE, i es podrà connectar amb la resta de dispositius Android.
L'objectiu de Google amb aquest nou sistema no és només el que puguem connectar-nos des del nostre mòbil a la nevera o al microones, De fet la idea és que també pugui ser utilitzat per qualsevol altre xip de baixa potència i consum, de manera que pugui ser utilitzat en qualsevol petit dispositiu, alguna cosa bastant important tenint en compte que avancem cap a un futur en el que tot està connectat amb tot.

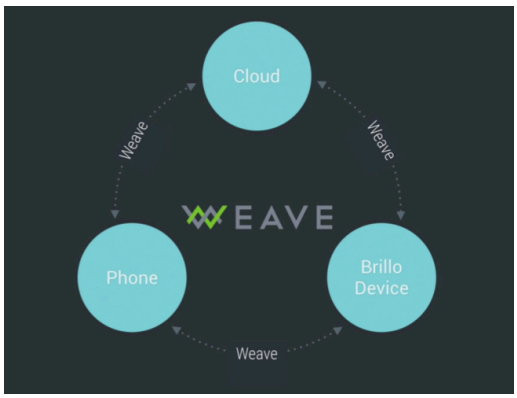


Fig. 128 Projecte Brillo O.S.

El que han creat des de Google, per a fer possible la interacció entre dispositius, és el llenguatge Weave, será una plataforma oberta que començarà a estar disponible pels desenvolupadors a finals d'any.

Aquesta plataforma neix per comunicar els nostres telèfons, el núvol i els dispositius que utilitzin Brillo, de manera que, per exemple, el pany de la nostra porta es pugui comunicar amb els llums perquè aquestes s'apaguin automàticament quan ens anem de casa, o fins i tot per poder encendre l'aire condicionat quan estem a punt de tornar a casa.